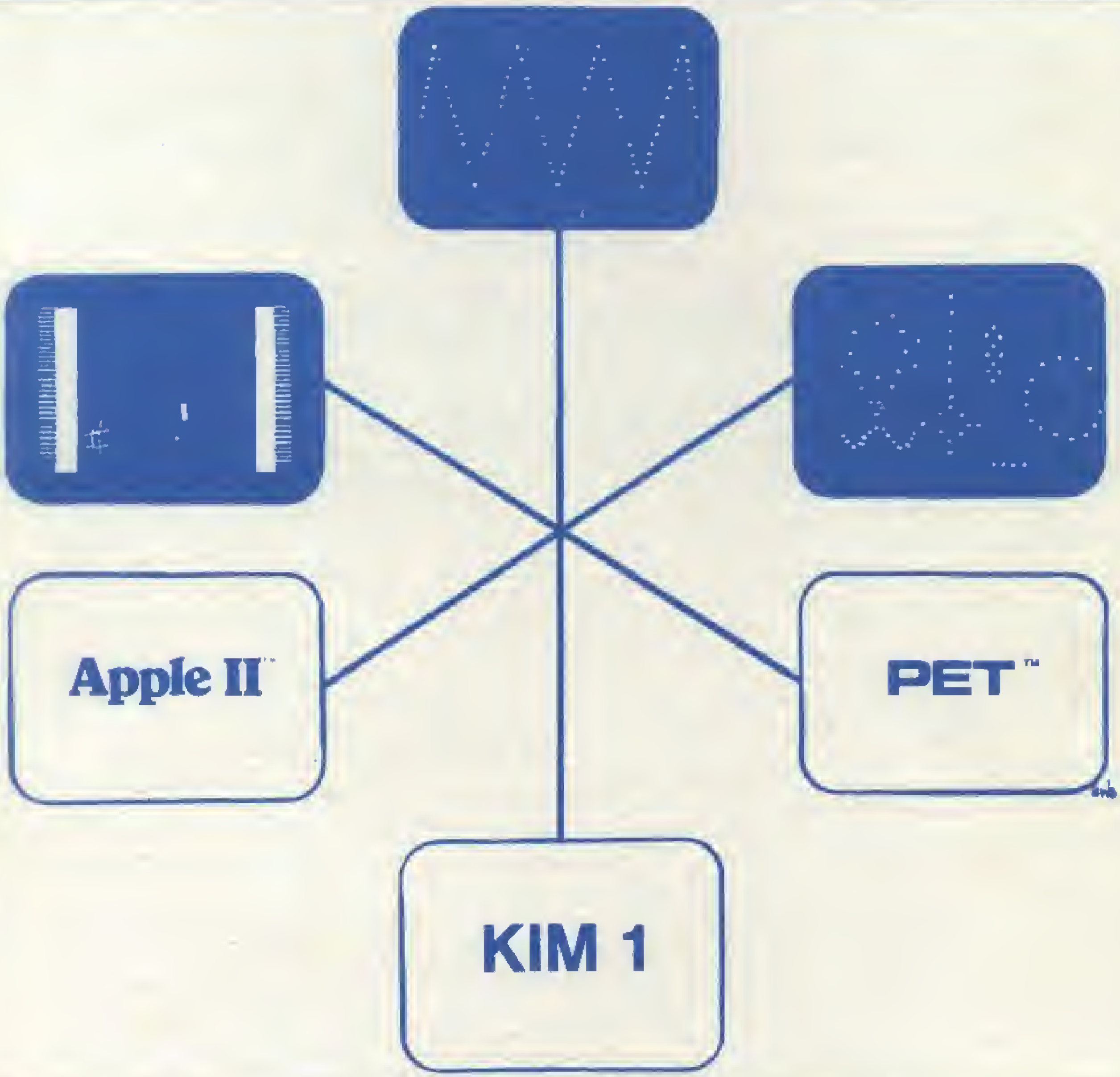
MICRO™

The Magazine of the APPLE, KIM, PET

and Other 6502 Systems



NO 8 **Dec 78 - Jan 79** $1.50

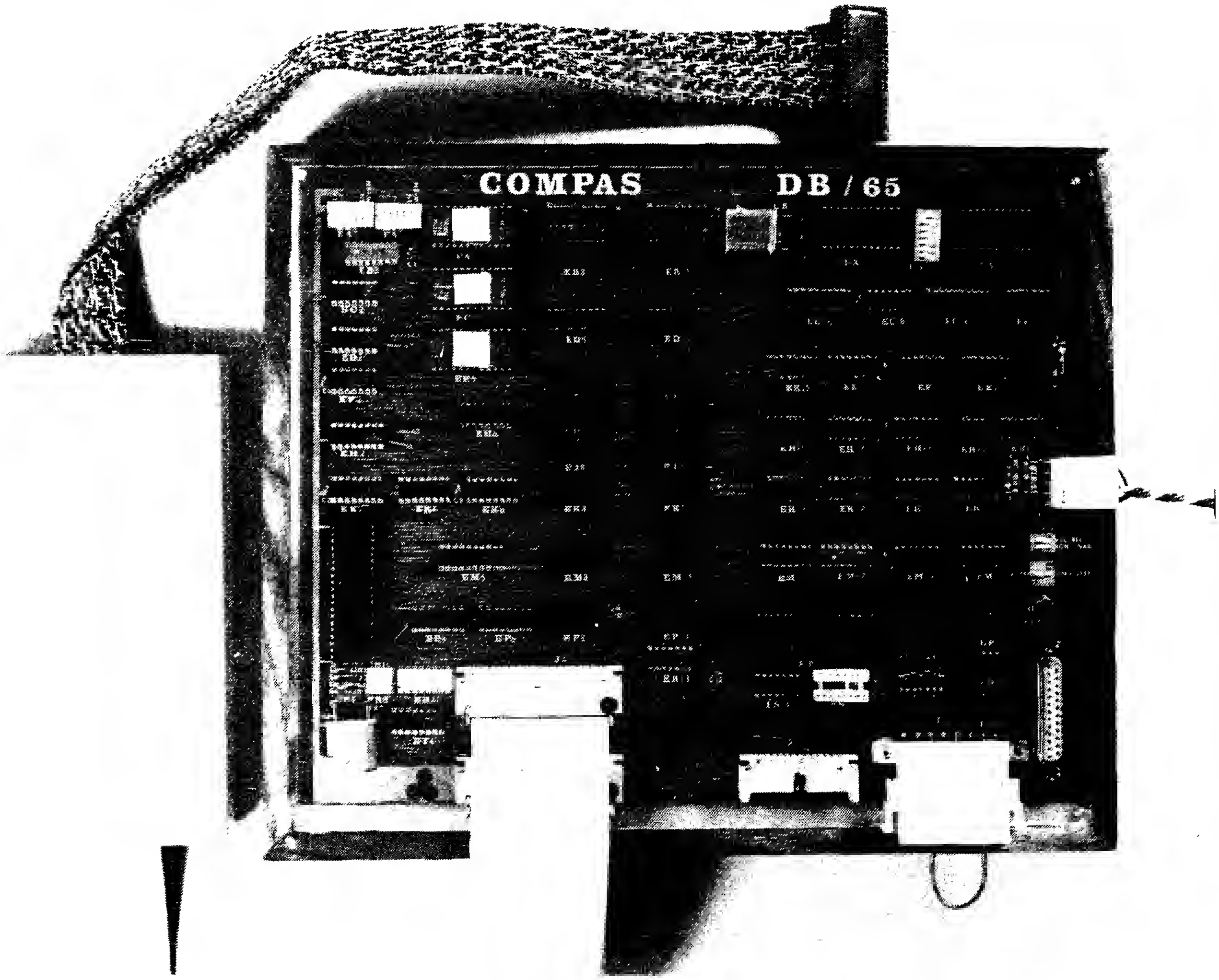
compas **microsystems**

P.O. Box 687 224 S.E. 16th Street Ames, Iowa 50010

515/232-8187

DB/65

A complete hardware/software debug system for the Rockwell, Synertek, MOS/Technology 6500 microprocessor family.



Features

* Standard in-circuit emulator
* Hardware breakpoints
* Single step mode
* Eight software breakpoints
* Real-time software breakpoints
* RS 232C or current loop terminals
* Symbolic disassembly of user program
* Serial/parallel load capability
* Program trace of instructions and registers
* Prom resident debug monitor
* Software history of instruction addresses
* 2K ram standard with sockets for additional 6K if required
* Scope sync output
* User NMI and IRQ vectors supported
* Write protect
* User program may reside in high memory

SINGLE QUANTITY PRICE = $1450

MICRO™

DECEMBER 1978/JANUARY 1979

ISSUE NUMBER EIGHT

Advertiser's Index

MICRO is published bi-monthly by:
The COMPUTERIST, Inc.,
P.O. Box 3, S. Chelmsford, MA 01824
Controlled Circulation postage paid at:
Chelmsford, MA 01824.
Publication Number: COTR 395770.
Subscription in US: $6.00/6 issues.

Please address all correspondence,
subscriptions, and address changes to:
MICRO
P.O. Box 3, S. Chelmsford, MA 01824

IN THIS ISSUE ...

This is the last bi-monthly issue of MICRO. Starting with the February 1979 issue, number 9, MICRO will be published monthly. The increase publication frequency is due to high volume and high quality of the articles being submitted for publication. Our backlog of good articles is growing too large. Also, with the addition of the Synertek SYM-1 and the Rockwell AIM 65, we anticipate a flood of new material to service these devices. The size and shape of MICRO will remain essentially unchanged: 8 1/2 x 11 format and 52 pages (or more) per issue. The subscription rate will remain the same: $1.00 per issue. Subscriptions will be accepted for any period of six issues or more. Another plus of monthly publication is that there will be a shorter delay between receipt of material and publication. This will permit us to print current club notes, special activity notices, and so forth.

Continuing his tutorial on "6502 Interfacing for Beginners", Marvin L. De Jong this month presents "Buffering the Busses". Earlier sections discussed the logic of the Address, Data and Control Busses. This article goes into some of the necessary detail on actually using these in real systema.

In the June/July 1978 issue of MICRO, Dr. Frank Covitz presented "LIFE for your PET". Now all of the Apple owners get an equal opportunity to play "LIFE" with Richard F. Suitor's "LIFE for your Apple" (A suggested title of 'LIFE IN your Apple' was rejected as implying worms!). This program combines a BASIC program to setup the initial pattern with assembly language code to perform the numerous tests and transformations. While it is okay to have fun and enjoy this program, you are expected to learn about using your display at the same time.

No one will mistake the article by Dr. L.S. Reich as a game. "Computer-Determined Kinetic Parameters in Thermal Analysis" presents a serious use for an Apple II in a lab analysis situation. This is definitely not a "beginners" article, but we hope it will help induce others to present some of their "real" uses for their microcomputer systems.

Alan K. Christensen shows how to overcome some shortcomings in using BASIC on the PET for real-time control with his "Continuous Motion Graphics of How to Fake a Joystick with the PET". In this article you will learn something about how the PET interpreter gets keyboard input and how your program can "hook" into this mechanism. The result is a keyboard style "joystick" which allows you to easily move around the display. A table is included which shows the relationship of the keycaps, screen value, and keyboard hex value. This table should be an aid in a variety of PET/Display oriented programs.

Powlette and Jeffery have updated the material presented by Marvin De Jong in the Dec 77-Jan 78 issue of MICRO with "Storage Scope Revisited". With a modified hardware circuit and a correction to the program, they produce results which are of quite high quality.

Rick Auricchio, to whom Apple owners are already in debt for his "An Apple II Programmer's Guide" in MICRO number 4 and "BREAKER: An Apple II Debugging Aid" in MICRO number 7, has now come up with "An Apple II Program Relocator" to further assist the Apple II community. This program, whose utility will be obvious to any programmer who does much in assembly language, also shows some techniques for using the SWEET-16 utility.

John Gieryic has wasted no time getting into action with his SYM-1 as evidenced by his need for a "SYM-1 Tape Directory" facility which he presents in his article. This complete program permits the user to examine his cassette tape to find what information is located on the tape. Since numerous calls are made to the SYM-1 monitor, it is a good guide to using monitor subroutines.

Jim Butterfield, widely known for his contributions to the KIM via "The First Book of KIM", has written a couple of programs which both aid and instruct the user of PET BASIC. One program allows a BASIC program to be searched for a particular data string with all lines which contain the string to have their line number printed. A second program permits a BASIC program to be resequenced, including fixing up GOTOs and other functions which reference the line numbers. His explanation of the workings of the programs will aid in the user's understanding of how BASIC is structured.

M. R. Connolly Jr. makes life easier for the Apple II user who is trying to work with the on-screen text by providing "An Apple II Page 1 Map" and a chart of the interpretation of values stored in the screen text buffer. Given this information, it becomes relatively easy to work on the display using PEEKs and POKEs.

... AND NOTES

"Attention SYM-1 and AIM 65 Users!!!! The San Fernando Valley KIM-1 Users Club is expanding its membership to include these two new and exciting microcomputer systems. We meet at 7:30 PM on the second Wednesday of the month at 20224 Cohasset No. 16, Canoga Park, CA 91306. Call Jim Zuber at (213) 341-1610 if you have any questions."

"This is to inform you of:

Apple Users Group
Neil D. Lipson
Phila. Area Comp. Soc. (PACS)
29 S. New Ardmore Avenue
Broomall, PA 19008
(215) 356-6183 (home)
(215) 825-3800 x278 (work)

For anyone owning an Apple computer."

"I am interested in starting an Apple User Group in the Cincinnati, OH area.. John B. Anderson, 5707 Chesapeake Way, Fairfield, OH 45014"

"Lincoln Computer Club is a non-profit school club that is made up of about 60 seventh and eighth grade students. We have a PET that we use for instruction and games. We would like to exchange programs with other PET users. Send a self-addressed, stamped envelope for info to:

PET Software Exchange
Lincoln Computer Club
Lincoln School
750 E. Yosemite
Manteca, CA 95336

MICROBES

Boy, is our face red. An entire chuck of code from "BREAKER: An Apple II Debugging Aid" ended up on the "cutting room floor". [MICRO 7:5] We apologize to the author, Rick Auricchio, and to anyone who has lost hair and/or sleep trying to get BREAKER working. The missing code is printed below. You can tell from the PC counter where it should be inserted into the original material. This is our biggest goof to date. We are moving to new quarters right now, and will have space to keep our microcomputers available for testing programs, so this should not happen again.

```
7E66    91 40              STAIY   A3L        STUFF ADDRESS INTO JMP
7E68    A5 2E              LDAZ    FORMAT     GET INSTRUCTION FORMAT
7E6A    C9 9D              CMPIM   X'9D'      IS FORMAT=BRANCH?
7E6C    F0 16              BEQ     ADDBRCH    =>YES. MORE TO DO
7E6E    A5 2F              LDAZ    LENGTH     LENGTH=1?
7E70    F0 0F              BEQ     CMDRET     =>YES. DONE
7E72    6A                 RORA               LENGTH=2?
7E73    B0 06              BCS     ADDLEN2    =>YES
7E75    A0 02              LDYIM   2          LENGTH=3;MOVE 3RD BYTE TO BTE
7E77    B1 3E              LDAIY   A2L        GET INST 3RD BYTE
7E79    91 40              STAIY   A3L         AND MOVE TO BTE
7E7B    A0 01     ADDLEN2  LDYIM   1          LENGTH=2;MOVE 2ND BYTE TO BTE
7E7D    B1 3E              LDAIY   A2L        GET INST 2ND BYTE
7E7F    91 40              STAIY   A3L         AND MOVE TO BTE
7E81    4C 69 FF  CMDRET   JMP     MON        DONE; BACK TO MONITOR!
                  *
                  * --- FOR BRANCHES, WE'VE GOTTA ADD A JMP FOR THE 'TRUE'
                  *       CONDITION (SINCE WE MOVED THE BRANCH 'WAY OUTA THE PROGRAM!)
                  *
7E84    A0 01     ADDBRCH  LDYIM   1          SET FOR 2ND BYTE
7E86    B1 3E              LDAIY   A2L        GET DESTINATION OFFSET
7E88    18                 CLC                 AND ADD 2 BYTES TO
7E89    69 02              ADCIM   2            CONSTRUCT ABS ADDRESS
7E8B    65 3E              ADCZ    A2L        ADD TO SUBJECT-INST ADDRESS
7E8D    85 3E              STAZ    A2L
7E8F    A5 3F              LDAZ    A2H        CARRY IT
7E91    69 00              ADCIM   0
7E93    85 3F              STAZ    A2H
7E95    EA                 NOP                (PLACE-HOLDER WASTE HERE)
7E96    A9 04              LDAIM   4          TRUE-BRANCH TO +4
7E98    91 40              STAIY   A3L        PUT INTO NEW OFFSET
7E9A    A0 07              LDYIM   7
7E9C    A5 3E              LDAZ    A2L        GET JMP ADDRESS
7E9E    91 40              STAIY   A3L        MOVE IT TO
7EA0    C8                 INY                 THE
7EA1    A5 3F              LDAZ    A2H          BTE FOR
7EA3    91 40              STAIY   A3L           THE 'TRUE' JMP
7EA5    B8                 CLV                SNEAKY BRANCH
7EA6    50 D9              BVC     CMDRET      TO EXIT
```

Henry Chow of Bloomfield Hills, MI pointed out the following typos in the "Design of a PET/TTY Interface" by Charles R. Husbands[MICRO 6:5].

LDA	COUNT	893	173	251	03
TAX		901	171		
INC	857	904	238	89	03
STA	857	927	141	89	03
LDA	SAD	951	173	79	232

It is very difficult for us to get listings of this sort correct. There are just too many ways to make mistakes, even with careful proofing. We are going to have to insist on computer generated listings for all articles from now on. If possible, authors should submit their source on cassette tape and let us list it on our own computers.

And now a first: A microbe in the 6502 Bibliography! Randall Julin writes that his article on the "Video Mixer" should have indicated "... video signals put out by the PET's Parallel User's Port, not the IEEE 488 bus."

6502 INTERFACING FOR BEGINNERS:

BUFFERING THE BUSSES

Marvin L. De Jong
Dept. of Math-Physics
The School of the Ozarks
Pt. Lookout, MO 65726

The address bus is the set of 16 conducting lines interconnecting the 6502 and numerous other integrated circuits in the computer system such as memory chips, PIAs, decoding circuits, etc. On my 8K memory board the address bus is connected to 64 memory chips. The address bus carries the addressing information from the 6502 to the other components in the system. It is, consequently, a one-way bus, in contrast to the data bus which carries signals both ways.

The control bus is a set of conductors which connect the 6502 control signals (0 , R/W, SYNC, RST, NMI, IRQ, RDY, and SO) with the other components in the microcomputer system. Some control signals originate in the 6502 and these are bussed to the system. Other control signals e.g. NMI and IRQ, originate somewhere in the system and are bussed to the 6502. None of the control signals use a bi-directional bus like the data bus.

Finally, the data bus is a set of 8 conductors connecting the 6502 and the other devices in the system. It presents a special problem because it is required to carry information two ways, hence the name "bi-directional data bus." On a WRITE command the data bus carries an 8-bit word (one bit on each line) from the 6502 to a memory location, while on a READ command the data bus carries information from a memory location to the 6502. On my 8K memory board each data line is connected to 8 memory chips.

WHY BUFFER?

There are two reasons for buffering uni-directional busses like the address bus and the control bus:

1. The address and control pins on the 6502 are rated to drive one standard TTL load. In any but the simplest computer system there will be heavier loading than this.

2. Every conductor including those which make up the busses has some capacitance. Capacitors require time to charge and discharge and "distort" rapidly changing waveshapes. Buffer chips can drive a much larger capacitance than the 6502, and consequently may be inserted to preserve the integrity of the waveshapes of the signals.

In addition, the data bus requires a special kind of buffer. Recall that the microprocessor is capable of reading data from any of 65,536 devices. But only one at a time, please. All the others should act as if they are not there, which means they should be disabled somehow. If two devices are both attached to a data pin, one trying to raise it to logic 1 and the other trying to lower it to logic 0, not even a prophet can predict the result. The third reason for buffering applies only to bi-directional busses and may be summarized:

3. Buffers must be capable of isolating the bus from all of the devices on the bus except those which have been addressed (for example, the 6502 and an input port) and between which data is being transmitted.

BUFFER/DRIVER CHIPS

We mentioned earlier that all the bus pins on the 6502 are rated to drive one standard 7400 series TTL load. This means that you could connect about four 74LS00 series chips to a bus line, but if you tried to hang additional chips on these lines the circuit would probably not operate. For the address bus and the control bus the solution is to connect the 6502 pins directly to two 7404 inverters (or 74LS04's). A 7404 can drive 10 standard TTL loads and about 40 LS loads, while a 74LS04 can drive 20 74LS00 series loads. This should provide adaquate drive for most systems, provided the bus length is not to great. If you have a KIM-1 schematic you will note that both R/W and 0 are buffered in this manner, but that none of the address lines are buffered because the KIM-1 system is small enough to not require buffering. However if you expand, the address lines will also require buffering. As an example, see KIM USER NOTES, Issue #7,8 where Jim Pollock gives a KIM to S-100 circuit.

There are other chips called Bus Buffers/Drivers which can be used either on uni-directional busses or the bi-directional data bus. They come in packages of four (quad), six (hex) or eight (octal) buffer/drivers to a chip. If you want to look up the specs on some of these chips here are a few of the more popular ones.

74LS125 quad	DM8093 quad
74LS126 quad	DM8094 quad
LS367 hex	DM8097 hex
8T97 hex	81LS97 octal

All of these except the 81LS97 are readily available (Jameco, Godbout, Jade, etc.). The only place I have been able to find 81LS97s is Hamilton-Avnet. They are a bit more expensive and come in a 20 pin package, but they are nice because they can handle eight lines. Note that we have already used the 74LS367 to buffer address lines. Refer to the last several columns of this feature.

The truth table and logic symbol for a typical buffer/driver are given in Figure 1. Carefully focus your beady eyes on the function of the G (gate) input.

Note that when G is low the output follows the input logic level. The device is then doing its thing, namely driving the particular bus line to which it is attached. The inversion circle indicates that the buffer/driver is active (works) when the gate signal is a logic 0. Some buffers have no inversion circles, and they will be active when the gate is at logic 1. Perhaps the most important feature is the third state of the output in the truth table, which we have labelled "disabled." When the gate is high the device behaves as if it were disconnected from the bus, that is just as if a switch in series with output were opened. This property is the reason for calling these devices "three-state buffer/drivers" or" or "TRI-STATE buffer/drivers." (TRI-STATE is a trademark of National Semiconductor.)

Figure 2 shows how an LS 125 might be used on the bi-directional data bus. Only two bus lines are shown for simplicity. During a WRITE instruction the R/W line is low, enabling the buffers which drive the signals from the 6502 to the external devices. The other buffers which drive the 6502 are disabled. Analyze what would happen if they weren't disabled! During a READ instruction the R/W line is high, it is inverted by the LS04, and it enables the buffers driving the signal from the external devices to the 6502.

The scheme shown in Figure 2 is not the only possibility. For example, the S-100 bus would not have pins 3 and 5 connected, nor pins 8 and 12 connected. Instead, the data bus is divided into two separate busses at this point. The bus lines connected to pins 3 and 8 become a "data out" bus, while the lines connected to pins 5 and 12 become a "data in" bus. I am not aware of all of the advantages and disadvantages of this scheme, so we will not pursue it further.

AN EXPERIMENT

Connect an LS125 as shown in Figure 3. Note that RESET will very likely cause all the LEDs to light. Now run the following program:

```
0000  4C 00 00          START   JMMP START
```

This is an infinite loop. Do not try to relocate the program or the experiment may not work. You should observe that the LEDs on D0 and D1 are off while the other two are one. Can you explain why before I do?

Analyzed by clock cycles the activity on the data bus may be summarized as follows:

The LEDs connected to D3 and D2 get a pulse once every three clock cycles, which the eye interprets as a continuous glow. Now connect the gates (pins 1,4,10,13) to +5V instead of ground. None of the LEDs light. Why?

AN OBSERVATION

Refer to Figure 1 in the "INTERFACING...." column in MICRO #7. The input port illustrates how a buffer/driver isolates the data bus. Note that the device select pulse is connected to the gate of the LS367. Thus, only when the address lines select the input port and the 6502 is in the READ state does the LS367 control the data lines. Otherwise it is disabled and the 6502 gets its data elsewhere.

The output port of the same circuit illustrates another point. Suppose we had say eight output ports. Data lines D0-D7 would each have eight LS inputs hanging on them, and the 6502 would probably be unable to drive them. The solution would be to buffer the data lines from the 6502 to the output ports. In this case one would probably connect the R/W line to the buffer/driver gates.

AN APPLICATION

Again refer to Figure 1 in this column in MICRO # 7. Recall that the data lines were to be connected to the D inputs of the LS75 to complete the output port, replacing the switch. A complete 8-bit output circuit, with buffering, is shown in Figure 4. The device select circuitry is not repeated here. Up to eight output ports can be implemented using the device select pulses from the LS138. All you have to have are LS 75s. The buffering shown in Figure 4 would be more than adaquate for eight ports.

The 8-bit port with LEDs attached can be used as a debugging tool among other things. At a point in a program where you suspect trouble, and want to see the STATUS REGISTER for example, put a BREAK command. The last thing on the stack after a break is the status register contents. So, the interrupt vector should point to a program which pulls the last word off the stack and loads it at the address of the output port, STA $800F. A little panel could be made which indicates LED goes with which flag.

The scheme just mentioned can obviously be varied to indicate the contents of any of the important registers. One could get very elegant and use four ports to indicate X, Y, accumulator and status register simultaneously. Better yet, use the information you have learned to display the contents of X,Y,A, and P while the computer is in the single-step mode.

What's next? I hope to go into a keyboard input port in a little more detail, then look at a memory interface, unless I get some other ideas that is. Anyway, you ought to step out from among the trees to get a look at the forest by taking a long and studied look at Figure 1.1 of the MOS TECHNOLOGY HARDWARE MANUAL, the first figure in the book. A lot of the ideas we have been discussing are summarized there in a diagram of the microcomputer system as a whole.

Parts list of components used for the experiments.

1	AP Circuit Board (holds 8, 16-pin DIPs)
1	coil, #22 wire
8	LEDs
1	Edge connector for KIM-1
1	74LS145
2	74LS138
1	74LS04
1	74LS367
2	74LS75
2	74LS125
1	74LS76
2	4.7K to 10K resistors
2	DIP switches

An LS125 and LS04 in a bi-directional data bus buffering circuit. Only two data lines are shown buffered. Four LS125s would be required for all eight data lines. In this scheme the "write" buffers and "read" buffers are alternately disabled by the R/W line. Sometimes they are also disabled by device select pulses.

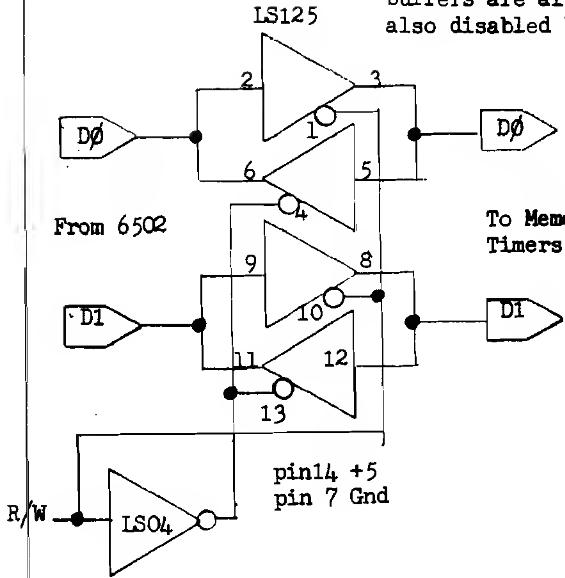


Figure 2.

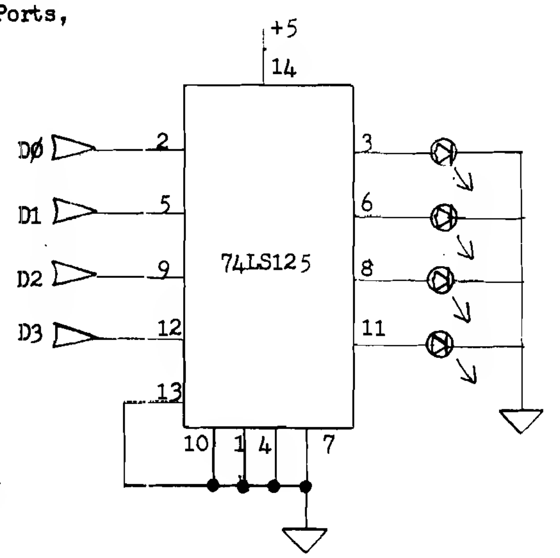


Figure 3.

Circuit to demonstrate data bus buffering. See text for details.

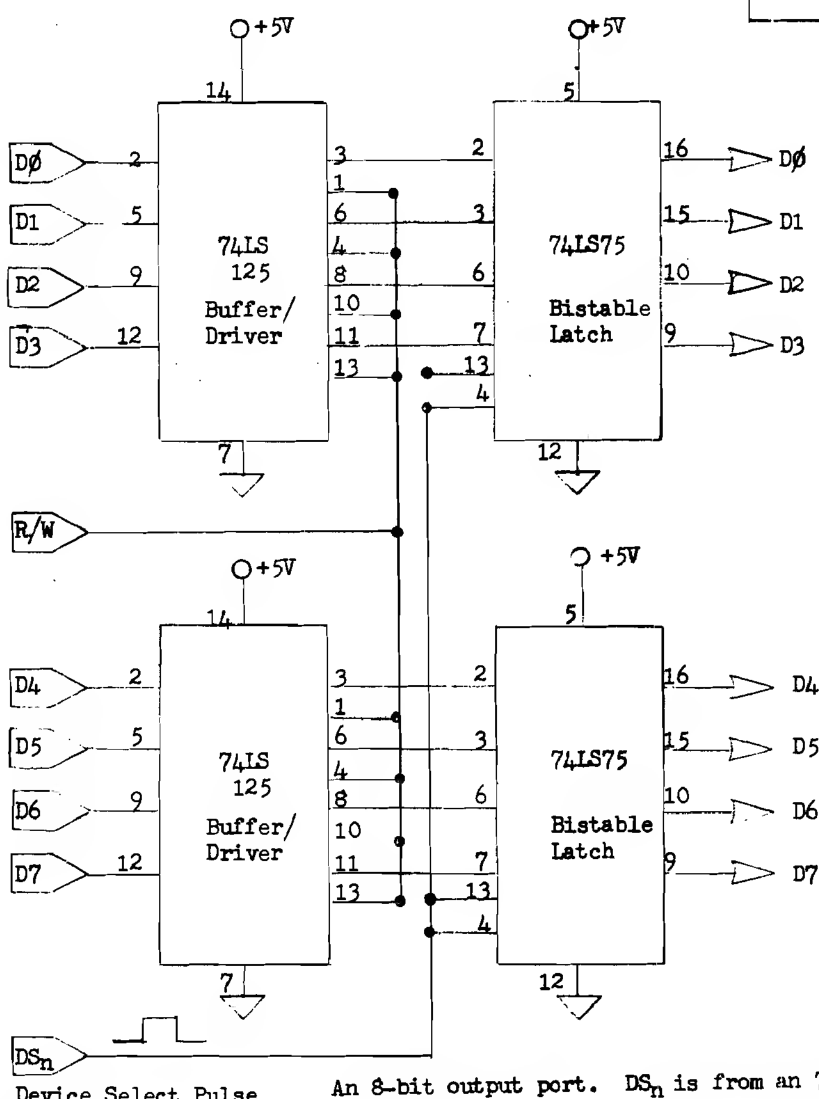


Figure 4.

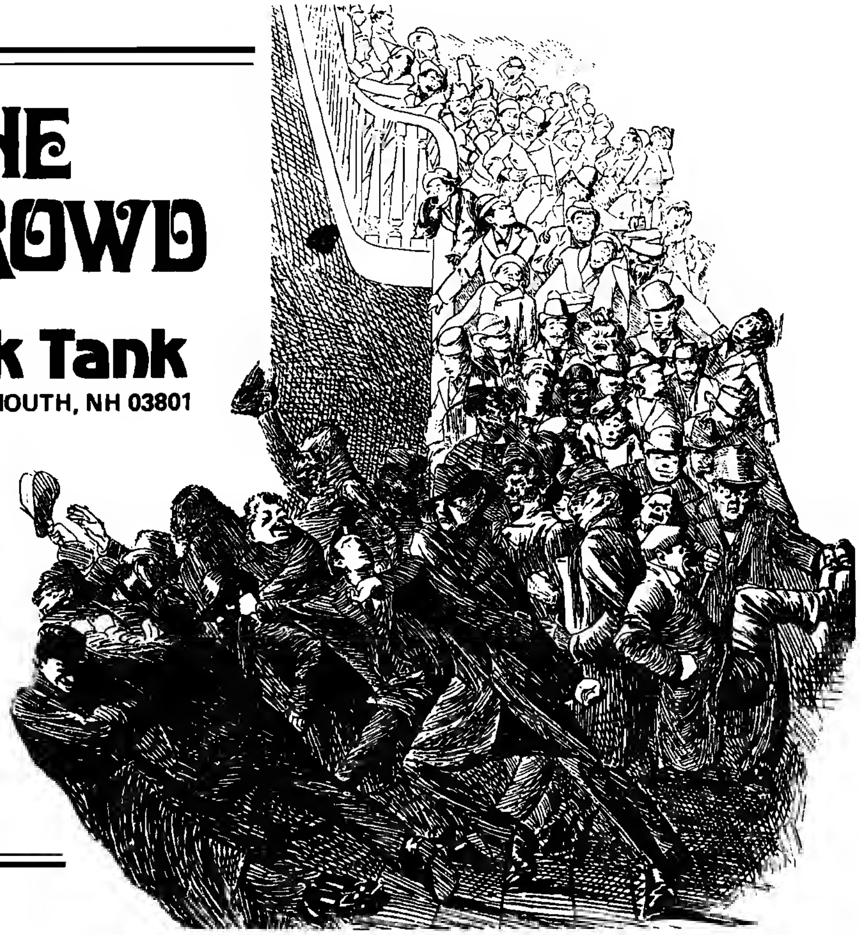
An 8-bit output port. DS_n is from an 74LS138 and LS04 inverter. The buffers could drive more ports.

FOLLOW THE CROWD

TO

The Think Tank

P O BOX 1199 • PORTSMOUTH, NH 03801



They're ALL rushing to buy the latest BOOKS on 6502 micro's!

Here are a few suggestions...

__The First Book of KIM__
by Butterfield, Ockers, & Rehnke
Step-by-step instructions on writing KIM programs , illustrating valuable techniques. Also, how to adapt the basic KIM-1 for home and business use. *$9.00*

__Programming a Microcomputer: 6502__
by Foster
How to program a micro in machine language. Well-written and clearly organized, suitable for self-study or classroom. Emphasis on problem ID, solution, and algorithms to solve the problem. *$9.00*

__Microcomputer Programming-6502__
by Zaks
How to program microprocessors with 6502: arithmetic, input-output, peripherals, interrupts. Educational text requiring no prior knowledge - for any 6502 micro. *$9.95*

Send your name, address and check or m.o. or charge to VISA or Master Charge (incl. number, exp. date, and interbank no.)

ASK FOR OUR HUGE MAIL ORDER CATALOGUE AVAILABLE IN LATE JANUARY, 1979

The Think Tank

P O BOX 1199 • PORTSMOUTH, NH 03801

DON'T FORGET

THESE ARE GREAT CHRISTMASS GIFT SUGGESTIONS!!!!

Stepping on board MICRO from PERSONAL COMPUTING has been quite an experience during the few short weeks I've been here. Thank goodness I have at least one microcomputer at home that is 6502-based! During the winter and spring of '79 I plan to add an APPLE II and a SYM to the OSI Challenger, making my intentions relatively solid in promoting the versatile world of t 6502 microprocessor.

I entered the micro world in April of '76 as an entrepreneur/hobbyist. My sixteen years in printing, advertising and publishing, along with several college courses and special academic projects in computer programming combine with a minimal writing/editorial background to round out my qualifications.

MICRO has several positive changes ahead over the next year, and I look forward to being a part of those changes.

Those of you who are manufacturers or software houses are reminded to submit (in OUR format) information on your products for listing in ourSoftware Catalog and Hardware Catalog.

The circulation has been steadily growing and although rates for advertising will be increasing, it won't be at the same rate as circulation.

Articles are starting to come in at a good pace, but we are always looking forward to new copy describing your 6502 application in hardware or software. Sophistication or simplicity, the article YOU write may bring in further inquiries or commentary from other 6502 users. This kind of dialogue proves to be very stimulating to most of us, so put that pen to paper and start writing!

MICRO will grow with its readers; just let u know what you want and we will try to please. We want to continue to be Number One in the 6502 world!

Gary

LIFE FOR YOUR APPLE

Richard F. Suitor
166 Tremont St.
Newton, MA 02158

A listing of LIFE for the APPLE II is described briefly here (see MICRO #5 for a pet version and discussions). Because my experience with generation time in BASIC paralleled Dr. Covitz', the generation calculations are in assembly language. The display is initiated in BASIC and the routines are called from BASIC, which will slow down the generation time if desired.

The entire (40x48) low resolution graphics display is used. An unoccupied cell is 0 (black). An occupied one is 11 (pink). During the first half of a generation, cells that will die are set to color 8 (brown). Those to be born are set to color 3 (violet). During this stage, bit 3 set indicates a cell is alive this generation; bits 0 and 1 set indicate a cell will be alive the next). During the second half (mop-up) part those with bits 0 set are set alive (color 11), the rest are set to zero.

The BASIC program allows one to set individual cells alive, and to set randomly 1 in N alive in a rectangular region. The boundries (X = 0 and 39; Y = 0 and 47) do not change, but may be initialized. At the start of the program, NO PADDLE INTERVAL? is requested. If during the program the paddle reads close to 255 (as it will if none is connected) the number input here will be used instead. Zero is fastest, several generations per second. Entering 200 gives a few seconds per generation.

When X and Y coordinates are requested, put in the coordinates for any cells to be set alive. A negative X terminates this phase. Setting X=N and a negative Y will initialize a rectangular region to 1 in N randomly occupied and terminate the initialization. The boundaries of the rectangular region must be input and may be anywhere in the full display. A glider gun can be fit vertically in the display. However, don't initialize for Y 40 (other than random) for the scrolling during initialization input will wipe it out.

Before RUNning the BASIC program, set LOMEM: 2500 to avoid overwriting the subroutines.

```
>LIST
    1 TEXT
    2 GEN=2088
    3 MOP=2265
    5 DIM A$(7)
    7 K1=1
    8 K2=1
   10 CALL -936: VTAB 5: TAB 9: PRINT
      "CONWAY'S GAME OF LIFE"
   30 VTAB 15: PRINT "INITIATE PATTERN
       BELOW.  X<0 WILL START"
   35 PRINT "THE LIFE PROCESS.  A Y<0
      WILL GIVE A"
   40 PRINT "RANDOM PATTERN WITH ONE I
      N X ALIVE"
   50 VTAB 22: INPUT "RETURN TO CONTIN
      UE",A$
   99 GOTO 1000
  100 REM
  102 POKE -16302,0
  103 GOTO 130
  104 FOR I=1 TO K3
  105 CALL GEN
  107 FOR K=1 TO K1: NEXT K
  110 CALL MOP
  112 FOR K=1 TO K2: NEXT K
  120 NEXT I
  130 REM
  131 KX= PDL (0)-10
  132 IF KX>240 THEN KX=KX1
  135 IF KX<0 THEN KX=0
  140 K1=KX*6
  150 K2=KX*2
  155 K3=500/(K1+50)+1
  160 GOTO 104
 1000 GR
 1010 CALL -936
 1020 INPUT "NO PADDLE TIME INTERVAL "
      ,KX1
 1100 COLOR=11: INPUT "INPUT X,Y "
      ,X,Y
 1105 IF Y<0 THEN 1800
 1110 IF X<0 OR Y<0 THEN 2500
 1120 IF X>39 OR Y>39 THEN 1100
 1130 PLOT X,Y: GOTO 1100
 1800 INPUT "X DIRECTION LIMITS "
      ,I1,I2
 1810 IF I1<0 OR I2>39 OR I1>I2 THEN
      1800
 1820 INPUT "Y DIRECTION LIMITS "
      ,J1,J2
 1830 IF J1<0 OR J2>47 OR J1>J2 THEN
      1820
 2000 CALL -936: GR
 2001 POKE -16302,0
 2002 CALL -1998
 2005 FOR I=I1 TO I2
 2010 FOR J=J1 TO J2: COLOR=11: IF
      RND (X) THEN COLOR=0
 2020 PLOT I,J
 2030 NEXT J
 2040 NEXT I
 2100 GOTO 100
 2500 POKE -16302,0
 2510 COLOR=0
 2520 FOR K=40 TO 47
 2530 HLIN 0,39 AT K
 2540 NEXT K
 2590 GOTO 100
 9000 END
```

```
!A

                0010 :LIFE ROUTINES
                0020 :ENTER AT GEN0 AND MOPO ALTERNATELY
                0030 :2088 AND 2265 DEC. RESP.
                0040 OLLN .DL 0002  OLD HORIZ LINE
                0050 NWLN .DL 0004  NEW LINE
                0060 SUM1 .DL 0006  # OF OCC. CELLS IN 3X3
                0070 SUM2 .DL 0007  1,2 FOR OLD,NEW
                0080 BUF1 .DL 0940  40 VERT. OCC. #S
                0090 BF1P .DL 0942
                0100 BF1M .DL 093F
                0110 BUF2 .DL 0970
                0120 BF2P .DL 0972
                0130 BF2M .DL 096F
0800  A505      0140 NXLN LDA *NWLN+01
0802  8503      0150      STA *OLLN+01
0804  A504      0160      LDA *NWLN
0806  8502      0170      STA *OLLN
0808  18        0180      CLC
0809  6980      0190      ADC 80
080B  8504      0200      STA *NWLN
080D  A505      0210      LDA *NWLN+01
080F  6900      0220      ADC 00
0811  C908      0230      CMP 08
0813  D00C      0240      BNE SAME
0815  A504      0250      LDA *NWLN
0817  6927      0260      ADC 27
0819  C952      0270      CMP 52
081B  1008      0280      BPL LAST
081D  8504      0290      STA *NWLN
081F  A904      0300      LDA 04
0821  8505      0310 SAME STA *NWLN+01
0823  18        0320      CLC
0824  60        0330 RTS1 RTS
0825  38        0340 LAST SEC
0826  B0FC      0350      BCS RTS1
                0360 :GENERATE BIRTHS(COLOR=3) & DEATHS(COL=8)
0828  20CA08    0370 GEN0 JSR INIT
082B  200008    0380 GEN1 JSR NXLN
082E  9001      0390      BCC GEN2
                0400 :ALL DONE IF CARRY SET
0830  60        0410      RTS
0831  A027      0420 GEN2 LDY 27
0833  98        0430      TYA
0834  AA        0440      TAX
                0450 :COMP VERT OCC #S
0835  A900      0460 GEN6 LDA 00
0837  994009    0470      STA BUF1,Y
083A  997009    0480      STA BUF2,Y
083D  B102      0490      LDA (OLLN),Y
083F  F00F      0500      BEQ GEN3
0841  1006      0510      BPL GEN7
0843  FE4009    0520      INC BUF1,X
0846  FE7009    0530      INC BUF2,X
0849  2908      0540 GEN7 AND 08
084B  F003      0550      BEQ GEN3
084D  FE4009    0560      INC BUF1,X
```

Note: The stars in the operand indicate zero page mode.

```
0850  B104    0570  GEN3 LDA (NWLN),Y
0852  F00F    0580       BEQ GEN5
0854  1003    0590       BPL GEN4
0856  FE7009  0600       INC BUF2,X
0859  2908    0610  GEN4 AND 08
085B  F006    0620       BEQ GEN5
085D  FE7009  0630       INC BUF2,X
0860  FE4009  0640       INC BUF1,X
0863  88      0650  GEN5 DEY
0864  CA      0660       DEX
0865  10CE    0670       BPL GEN6
0867  A026    0680       LDY 26
0869  18      0690       CLC
086A  AD6709  0700       LDA BUF1+27
086D  6D6609  0710       ADC BUF1+26
0870  8506    0720       STA ◆SUM1
0872  AD9709  0730       LDA BUF2+27
0875  6D9609  0740       ADC BUF2+26
0878  8507    0750       STA ◆SUM2
              0760  :COMP OCC #S IN 3X3 & CHANGE COLOR
087A  18      0770  GNLP CLC
087B  A506    0780       LDA ◆SUM1
087D  793F09  0790       ADC BF1M,Y
0880  38      0800       SEC
0881  F94209  0810       SBC BF1P,Y
0884  8506    0820       STA ◆SUM1
0886  C903    0830       CMP 03
0888  F00E    0840       BEQ GEN9
088A  9004    0850       BCC GEN8
088C  C904    0860       CMP 04
088E  F00E    0870       BEQ GN10
0890  B102    0880  GEN8 LDA (OLLN),Y
0892  F00A    0890       BEQ GN10
0894  298F    0900       AND 8F
0896  5004    0910       BVC GN16
0898  B102    0920  GEN9 LDA (OLLN),Y
089A  0930    0930       ORA 30
089C  9102    0940  GN16 STA (OLLN),Y
089E  18      0950  GN10 CLC
089F  A507    0960       LDA ◆SUM2
08A1  796F09  0970       ADC BF2M,Y
08A4  38      0980       SEC
08A5  F97209  0990       SBC BF2P,Y
08A8  8507    1000       STA ◆SUM2
08AA  C903    1010       CMP 03
08AC  F00E    1020       BEQ GN12
08AE  9004    1030       BCC GN11
08B0  C904    1040       CMP 04
08B2  F00E    1050       BEQ GN13
08B4  B104    1060  GN11 LDA (NWLN),Y
08B6  F00A    1070       BEQ GN13
08B8  29F8    1080       AND 0F8
08BA  5004    1090       BVC GN15
08BC  B104    1100  GN12 LDA (NWLN),Y
08BE  0903    1110       ORA 03
```

```
08C0  9104     1120  GN15 STA (NWLN),Y
08C2  88       1130  GN13 DEY
08C3  F002     1140       BEQ GN14
08C5  10B3     1150       BPL GNLP
08C7  4C2B08   1160  GN14 JMP GEN1
08CA  A904     1170  INIT LDA 04
08CC  8505     1180       STA *NWLN+01
08CE  A900     1190       LDA 00
08D0  8504     1200       STA *NWLN
08D2  8D6809   1210       STA BF1P+26
08D5  8D9809   1220       STA BF2P+26
08D8  60       1230       RTS
               1240  ;MOP UP, IF COLOR AND 3 =0, REMOVE(COL=0)
               1250  ;OTHERWISE, ALIVE (COL=11)
08D9  20CA08   1260  MOP0 JSR INIT
08DC  200008   1270  MOP1 JSR NXLN
08DF  9001     1280       BCC MOP2
08E1  60       1290       RTS
08E2  A027     1300  MOP2 LDY 27
08E4  B102     1310  MOP3 LDA (OLLN),Y
08E6  F00A     1320       BEQ MOP5
08E8  297F     1330       AND 7F
08EA  C910     1340       CMP 10
08EC  3002     1350       BMI MOP4
08EE  0980     1360       ORA 80
08F0  9102     1370  MOP4 STA (OLLN),Y
08F2  B104     1380  MOP5 LDA (NWLN),Y
08F4  F00A     1390       BEQ MOP7
08F6  29F7     1400       AND 0F7
08F8  6A       1410       ROR
08F9  9002     1420       BCC MOP6
08FB  0904     1430       ORA 04
08FD  2A       1440  MOP6 ROL
08FE  9104     1450       STA (NWLN),Y
0900  88       1460  MOP7 DEY
0901  F0D9     1470       BEQ MOP1
0903  10DF     1480       BPL MOP3
               1490       .EN
```

```
SYMBOL TABLE
OLLN    0002
NWLN    0004
SUM1    0006
SUM2    0007
BUF1    0940
BF1P    0942
BF1M    093F
BUF2    0970
BF2P    0972
BF2M    096F
NXLN    0800
SAME    0821
RTS1    0824
LAST    0825
GEN0    0828
GEN1    082B
GEN2    0831
GEN6    0835
GEN7    0849
GEN3    0850
GEN4    0859
GEN5    0863
GNLP    087A
GEN8    0890
GEN9    0898
GN16    089C
GN10    089E
GN11    08B4
GN12    08BC
GN15    08C0
GN13    08C2
GN14    08C7
INIT    08CA
MOP0    08D9
MOP1    08DC
MOP2    08E2
MOP3    08E4
MOP4    08F0
MOP5    08F2
MOP6    08FD
MOP7    0900

!
```

COMPUTER-DETERMINED KINETIC PARAMETERS IN THERMAL ANALYSIS

Dr. L.S. Reich
3 Wessman Drive
West Orange, NJ C7052

INTRODUCTION

Two techniques employed in thermal analysis which are popular with chemists, chemical engineers, and other scientists studying the thermal degradation of various materials, e.g., teflon, are thermogravimetric analysis (TG) and differential thermal analysis (DTA). An important aspect of thermal analysis is the quantitative estimation of kinetic parameters for the material being degraded such as, activation energy, E (cal/mole), and reaction order, N.

Prior to the advent of computers (and programmable calculators), there was an understandable tendency to avoid accurate, sophisticated (but time-consuming and laborious) methods of data analysis to obtain values of E and N. Graphical methods were employed to a large extent. Recently, the author reported an accurate, sophisticated method (no graphics need be involved) whereby raw conversion-temperature data could be rapidly analyzed by a computer (also, but more laboriously by a programmable calculator, e.g., HP97) to yield values of E and N (Thermochim Acta, 24, 9 (1978); ibid., 25, 367 (1978). (In these reports, there was no description of the computer program used.) By employing an Apple II computer with Applesoft II Basic (20K) and the program listed in this article (ca. 10-11K free bytes required depending upon the amount of data entered), the time required to estimate E and N by the reported method, for the thermal degradation of teflon via TG (as an example), beginning with data entry to the display of preliminary results followed by one iteration to obtain accurate final results, was only ca. 4 min.

In this article will be described the computer program which can be used with the previously reported method for the estimation of E and N from data derived by thermal analysis.

SOME BACKGROUND INFORMATION

In the report previously mentioned (loc. cit.), the following expression was derived (can be used for TG and DTA):

$$\frac{E}{R} = \mathrm{LOG}\left[\left(\frac{1-(1-\alpha_1)^{1-N}}{1-(1-\alpha_2)^{1-N}}\right)T(1)\right]U(1)$$

where, $T(1)=(T_2/T_1)^2$; $U(1)=T_1 T_2/T_1 - T_2)$; R = gas constant (1.9872 cal/deg-mole); α denotes fractional conversion; α_1 corresponds to temperature (K), T_1, etc.

For two pairs of given values of α and T, i.e., α_1, T_1 and α_2, T_2, values of E/R can be calculated from the above expression for various arbitrarily selected values of N. However, assuming uniqueness, only one pair of E, N values will be significant. By using other pairs of α and T values, other sets of values of E and corresponding N will be obtained. In all these sets there should be only one pair of E, N values in common. However, such values would rarely, if ever, be expected to be exactly equal in practice due to experimental limitations such as, sample impurities, heat transfer effects, etc. Therefore, these values were taken to be those whose mean deviation (MD) was the least of all of the MD's obtained for all the sets of values obtained. Although the above expression does not apply when N is exactly equal to unity it is rare in practice for reactions to be exactly first-order and hence this equation is considered to be of general validity. When values of N close to unity are used, the value of N may be set equal to 1.0001, for example, in order to avoid the error message, "division by zero error" (this technique was employed in this paper). Once E and N have been evaluated, another parameter, the pre-exponential factor, may also be evaluated. This factor was not considered in this paper.)

THE PROGRAM

The program listed has the following limitations. The values of N should not be greater than 3 (termolecular reactions are extremely rare, if they occur at all, during thermal degradations). Also, the data which is entered in line #200, is limited to ca. 44 data pairs (most raw data do not contain so many data pairs of conversion-temperature, but if necessary, the number of such pairs may be increased by adjusting the DIM statement for A and T in line #7). The value of N cannot be equal to 1 exactly, otherwise an error message will result. This may be circumvented by using N=1.0001, for example. The Apple II screen will only accommodate ca. 6 columns of E/R values (6|N-values). Nevertheless, more than 6|N-values may be used, even though the display may appear confusing. about 10-11K free bytes will be required for the program, depending upon the amount of data entered. Further, since subscripted variables must contain integer subscripts and since N usually varies from .5-2, reaction orders are given as N x 100. This increases the DIM statement and consequently the number of bytes required by the program.

In the program itself, explanatory REM statements are to be found in line #'s 8, 47, 70, 80, 135, and 138. Prior to running the program, data pairs of conversion-temperature (K) must be entered (see line #200). Then line #5 must be properly adjusted. In this line # (see line #2) Y denotes the initial order (x100), Z denotes the final order (x100), and the increment is given by V (x100). Thus, for the teflon data depicted in line #200 (from TG), the initial order will arbitrarily be .86 (Y=86) and the final order 1.11 (z=111) while the increment will be .05 (V=5) to yield 6 N-values. The preliminary results obtained using these values were: E/R = 33091 ± 872 for N = 1.01 ± .05. Since the value of N was now established as ca.1 more refined values were obtained using Y = 97.01, Z = 101.01, V = 1 (the .01) was used to avoid a division by zero error message). Final values now were: E/R=32792 ± 822 for N=.98 ± .01.

As stated in line #8, line #'s 10-40 are used to form an M x J array of conversion-temperature, A(M,J). Line #'s 48-76 allow the calculation of E/R (Z(N)), according to the expression previously mentioned, for various orders and for various conversion-temperature data pairs. Also, S(N) (line #70) is the summation of all Z(N) (E/R) values for any particular order, N, and is subsequently used to obtain the average E/R value and its MD for a particular N (see line # 125). Line #'s 84-110 allow the determination of the sum of absolute differences, D(N), between E/R values and the average E/R value for a particular value of order, N. The average E/R value and its MD are calculated for a particular N in line #125. Finally, line #'s 140-165 allow the determination of the average E/R value that corresponds to the minimum MD at a certain order N. Line #'s 139 and 160 are used to estimate the value of N which corresponds to the "most probable" E/R value. In line #175, the most probable E/R value (minimum MD), its MD and corresponding N are printed. Along with the program listing are given results of an actual run using the teflon data in line #200 obtained by means of TG.

PROGRAM LISTING

```
1  PRINT "THIS PROGRAM ESTIMATES E/R VALUES FROM TG/DTA
   DATA OF CONVERSION VS. TEMPERATURE (K). THE PROGRAM
   DOESN'T APPLY FOR REACTION ORDERS > 3."
2  PRINT"IN LINE # 5, Y= INITIAL ORDER (x 100), Z= FINAL
   ORDER (x 100), WHILE THE INCREMENT IS GIVEN BY V (x 100)."
3  PRINT"FOR EACH RUN, THE VALUES IN LINE # 5 WILL PROBABLY
   NEED ADJUSTMENT. ABOUT 10-11K FREE BYTES WILL BE REQUIRED."
4  PRINT"WHEN DATA PAIRS OF CONVERSION-TEMP (K) HAVE BEEN
   ENTERED AND LINE # 5 HAS BENN ADJUSTED AND YOU ARE READY,
   TYPE 'CONT' " : PRINT"REM STATEMENTS ARE IN LINE #'S 8,
   47,70,80, 135, 138." : STOP
5  PRINT: Y= 86 : Z= 111 : V=5
7  DIM S(310), D(310), A(44,2), Z(310), U(44), T(44), C(310)
8  REM  LINE #'S 10-40 FORM ARRAY A(M,J) OF CONVERSION-
   TEMP DATA
10  FOR M= 1 TO 50
15  FOR J= 1 TO 2
20  READ A(M,J)
30  IF A(M,1)= 0  THEN 40
35  NEXT J,M
40  M= M - 1
42  PRINT" E/R VALUES OF REACTION ORDERS, N (x 100):"
43 PRINT
45  FOR K= Y TO Z  STEP V: PRINT "N= "K" "; : NEXT
46  PRINT
47  REM  LINE #'S 48-76 ALLOW THE CALCULATION OF Z(N) (E/R)
    FOR VARIOUS ORDERS AND FOR VARIOUS CONVERSION-TEMP
    DATA PAIRS
48  FOR  I= 1 TO M-1
50  T(I)= (A(I+1,2))^2/(A(I,2))^2
```

```
55  U(I)= A(I,2)* A(I+1,2)/(A(I,2)-A(I+1,2))

57  FOR N= Y  TO  Z STEP V

60  Z(N)= LOG((1- (1- A(I,1))^(1- (N/100)))* T(I)/
    (1- (1- A(I+1,1))^(1- (N/100))))* U(I)

65  PRINT  INT(Z(N));" ";

70 S(N)= S(N) + Z(N) : REM  S(N) IS SUM OF ALL Z(N) (E/R)
   VALUES FOR ANY PARTICULAR ORDER, N

72  NEXT N

74  PRINT

76  NEXT I

78  PRINT: PRINT "PRESS A KEY TO CONTINUE!"; : GET A$:PRINT

80  REM  LINE #'S 84-110 ALLOW DETERMINATION OF SUM OF
    ABSOLUTE DIFFERENCES (D(N)) BETWEEN E/R VALUES AND THE
    AVERAGE E/R VALUE FOR A PARTICULAR VALUE OF ORDER, N

84  FOR I= 1 TO M- 1

95  FOR N= Y TO Z  STEP V

100 Z(N)= LOG((1- (1- A(I,1))^(1- (N/100)))* T(I)/
    (1- (1- A(I+1,1))^(1- (N/100))))* U(I)

105  D(N)= D(N)+ ABS(Z(N)- (S(N)/(M- 1)))

110  NEXT N,I

115  PRINT

117  PRINT "AVG. E/R VALUES AND THEIR MEAN DEVIATIONS FOR
     VALUES OF ORDERS (N x 100): N= "Y" TO "Z" , INCREMENT "
     V" ARE RESPECTIVELY: "

118  PRINT

120  FOR W= Y TO Z  STEP V

125  PRINT S(W)/(M- 1)" + OR - "D(W)/(M- 1)

127  PRINT

130  NEXT W

134  PRINT : PRINT "PRESS A KEY TO CONTINUE!"; : GET A$: PRINT

135  REM  LINE #'S 140-165 ALLOW DETERMINATION OF THE E/R
     VALUE THAT CORRESPONDS TO THE MINIMUM MEAN DEVIATION
     AT A CERTAIN VALUE OF ORDER, N

138  REM  LINE # 139 ALONG WITH # 160 ARE USED TO DETERMINE
     VALUE OF ORDER, N, CORRESPONDING TO THE 'MINIMUM'
     E/R VALUE

139  FOR J= Y TO Z  STEP V : C(J)= J : NEXT

140  FOR W= Y TO (Z- V)  STEP V

145  FOR U= (Y+ V) TO Z  STEP V
```

```
147  IF D(W)<D(U)  THEN 165
155  Q= D(W): R= S(W): D(W)= D(U): S(W)= S(U): D(U)= Q: S(U)=R
160  E= C(W): C(W)= C(U): C(U)= E
165  NEXT U,W
170  PRINT: PRINT "IF ABOVE VALUES HAVE A MINIMUM, THE MOST
     PROBABLE VALUE OF E/R, ITS MEAN DEVIATION, AND ORDER, N,
     ARE RESPECTIVELY:"
175 PRINT : PRINT TAB(5); INT((S(Y)/(M - 1)) + .5)" + OR - "
    INT((D(Y)/(M - 1)) + .5)" FOR A VALUE OF N= "C(Y)/100" +
    OR - "V/100
200  DATA .016,773.2,.087,803.2,.216,823.2,.489,843.2,.663,
          853.2,.826,863.2
500  DATA 0
```

RESULTS FROM A RUN USING TEFLON DATA (FROM TG)

COMMAND 'RUN' ——> STATEMENTS IN LINE #'S 1-4 and "BREAK IN 4"

COMMAND 'CONT' ——>

"E/R VALUES OF REACTION ORDERS, N (x 100):

N= 86	N= 91	N= 96	N=101	N=106	N=111
34137	34176	34214	34253	34292	34331
30533	30659	30784	30910	31036	31162
32526	32891	33259	33629	34003	34379
30958	31682	32416	33162	33918	34685
29959	31110	32289	33498	34735	36001

PRESS A KEY TO CONTINUE! "

COMMAND: KEY PRESSED TO ——>

"AVG. E/R VALUES AND THEIR MEAN DEVIATIONS FOR VALUES OF ORDERS (N x 100): N= 86 TO 111, INCREMENT 5 ARE RESPECTIVELY:

31623.193 + OR - 1367.13721
32103.8155 + OR - 1144.06226
32593.017 + OR - 915.268713
33090.8084 + OR - 872.206769
33597.1905 + OR - 1024.30262
34112.159 + OR - 1179.69265

PRESS A KEY TO CONTINUE! "

COMMAND: KEY PRESSED TO ⟶

" IF ABOVE VALUES HAVE A MINIMUM, THE MOST PROBABLE VALUE OF E/R, ITS MEAN DEVIATION, AND ORDER, N, ARE RESPECTIVELY:

33091 + OR - 872 FOR A VALUE OF N= 1.01 + OR - .05 "

Another run was made using a smaller increment, V= 1, and Y= 97.01, Z= 101.01 (the .01 avoids a "division by zero error" message) to yield the more accurate final result:

" 32792 + OR - 822 FOR A VALUE OF N= .9801 + OR - .01 "

These results are in excellent agreement with results obtained from the same data by other methods (non-computer) which were laborious and time-consuming.

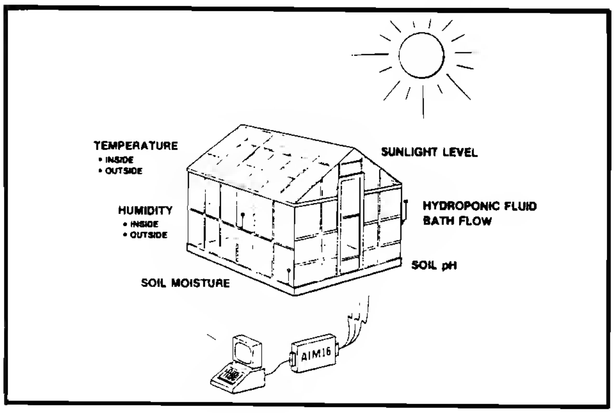
DAM YOUR PET
DAM YOUR TRS-80
DAM YOUR KIM
DAM YOUR . . .

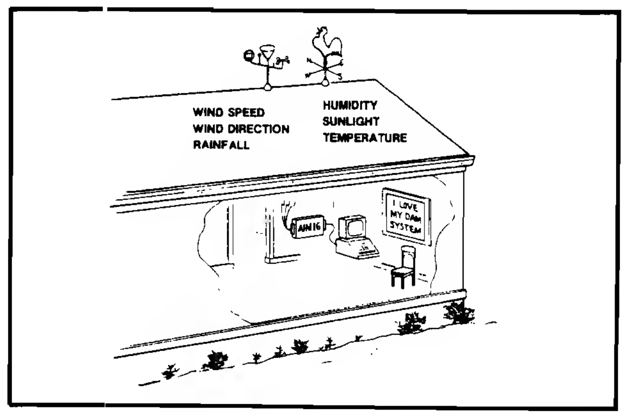
MEASURE - RECORD - CONTROL

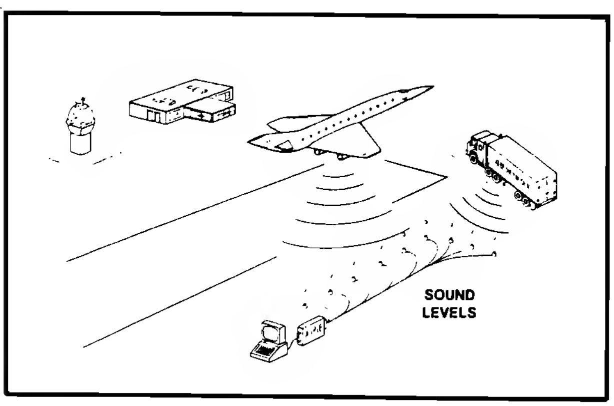
- TEMPERATURE
- DIRECTION
- PRESSURE
- LIGHT LEVELS
- db
- POLLUTION CONTROLS
- DARKROOMS
- HUMIDITY
- LIGHT
- ENERGY CONSERVATION EQUIPMENT
- GREENHOUSES
- SPEED
- WEATHER STATIONS
- NOISE POLLUTION
- pH
- EARTHQUAKE TREMORS
- VELOCITY
- ACCELERATION

DATA
ACQUISITION by CmC
MODULES

NOW YOUR COMPUTER CAN LISTEN TO THE REAL WORLD. YOU GET 16 8 BIT ANALOG INPUTS WITH OUR AIM16.







?

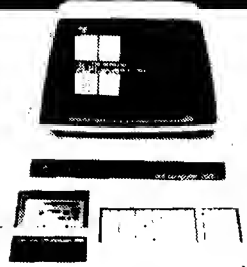
CONNECTICUT microCOMPUTER
150 POCONO ROAD · BROOKFIELD, CONNECTICUT 06804
(203) 775 9659

DAM SYSTEMS

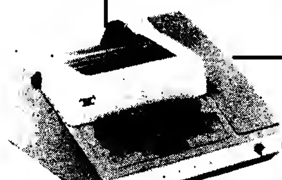
BUY FROM THE BIGGEST

NCE/CompuMart

1250 North Main Street, Department MJC8
P.O. Box 8610 Ann Arbor, Michigan 48107



PET
$795



THE PET CONNECTION

Networks Bidirectional RS-232 Interface
- Single Port $240.00
- Dual Port $280.00
- Cabinet $ 35.00

Networks Modem for PET
- Attaches to DAA $320.00
- Cabinet $ 35.00

PET with 8K user memory $795.00
Second Cassette drive $99.95
RS-232 Printer Adapter — assembled board $98.50
With power supply, case & connector $169.00
Beeper — PET controlled beep $24.95

Video Buffer — add a conventional monitor $29.95

EXPANDAPET - Economical 16K memory expansion for the PET (includes 1 daughter board) $438.00
without daughter board $395.00

Hams! Add a PET to your rig.

Computerize your station with Microtronics M-65 Ham Interface and included software. You simply plug the unit into your PET and you're ready to display, transmit and receive either CW or RTTY at rates up to 100 WPM.

ASSEMBLED KITS
$99.95

SECOND CASSETTE FOR PET-$99.95

This peripheral from Commodore gives you the ability to use one tape deck for input and one for output copy files etc.

SELECTRIC-BASED TERMINAL FOR PET

$895

We have interfaced our TC-71 to the PET. Use with the IEEE-488 bus as a printer or with the VIA User Port for bidirectional I/O. Ideal for Word Processing applications.

Sale price if bought with PET $795

LOW-COST PRINTER FOR PET

This is a completely refurbished Teletype KSR-33 terminal ready to attach to your PET's interface to use as a printer.

INTRODUCTORY PRICE $395

READER/PUNCH

8 LEVEL PAPER TAPE/READER PUNCH

USED (90 day warranty) $175.00
Documentation Package $15.00

NEW ASCII INTERFACES for Reader/Punch

30 Day delivery on Interfaces

Boards & Cables
- Parallel $220
- RS232 Serial $350

Board, Cables, Power Supply
- Parallel $490
- Serial $615

Board, Cables, Power Supply, Enclosure
- Parallel $575
- Serial $780

KIM-1

The original fully expandable single board computer

Assembled **$179**
with power supply **$209**

EXPAND YOUR KIM SYSTEM

KIMSI
KIM to S-100 Board

Assembled $169
Connector set $15

KIM4
Expansion board $119

MICRO TECH. POWER SUPPLY
Assembled $34.95

KIM ENCLOSURE

Give your KIM a professional appearance. No alterations required, sturdy Kydex 100 Black **$23.50**

WE WANT TO BUY USED MICROCOMPUTERS

Any Quantity

Urgently needed:
- Peripherals
- Manufacturers surplus
- Closed stores stock

CASH AVAILABLE

Call Today: (313) 994-3200

EXIDY SORCERER $895

- Z 80 Processor
- Full size ASCII keyboard
- User definable characters
- Up to 32K on board RAM
- Interchangeable ROM PAC's
- 30 x 64 display
- Resident 4K Monitor ROM
- Dual Cassette I/O
- Serial and Parallel I/O
- Complete Operators Manual
- Only $895

Sorcerer 8K	$895
Sorcerer 16K	$1150
Sorcerer 32K	$1395
16K Expansion	$299

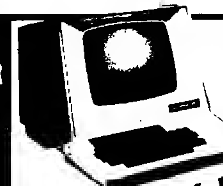
Single board VIDEO TERMINAL by XITEX

SALE!

16 Ln x 64 chr display, 128 printable characters, software UART (ASCII or Baudot), 3 serial interfaces & cassette I/O, true composite video, TTL compatible, easy power connection.

Assembled & tested **NOW $187**

Hazeltine 1400



IN STOCK NOW

Only **$749.00**

Now a 5th Generation in Terminals:

Introductory Sale Price — Immediate Delivery

- 24 x 80 display
- RS-232C interface
- 110 to 9600 Baud
- 12" CRT Monitor
- Remote screen control
- Uses 8048 microprocessor
- State-of-the-art design

A full capability terminal completely assembled with a 90 day warranty for less than you'd pay for a kit.

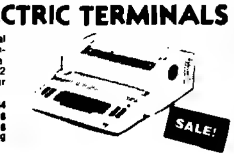
Compucolor II

NEW!

Model 3 8K RAM $1495
Model 4 16K RAM $1795
Model 5 32K RAM $2395

- Beautiful 128 x 128 color graphics
- 4K Graphics RAM
- 16K ROM with Disk BASIC
- Built-in 51K mini floppy
- RS 232 interface
- Full size keyboard
- Preprogrammed diskettes avail

TC-71 SELECTRIC TERMINALS



Top-quality printing terminal with RS-232, PTTC code. Includes desk, 130 char/line. ASCII version employs 6802 for code conversion. See our catalog for details.

ASCII	$884
Refurbished	$698
Working	$498
As-Is	$339

SALE!

SALE **ASCII INTERFACE FOR TC-71**
30 day delivery Normally $199 NOW **$169**
Acoustic Coupler for use with ASCII TC-71 **$159**

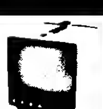
SYM-1

From Synertek **$269.00**

- 28 Double function keypad including 10 to 13 special functions
- Easy to view 6-digit hex LED display
- KIM-1 hardware compatibility
- 6512 based system
- 1K byte ROM resident monitor and operating system
- Single 5 volt power
- 4 Kbytes of 2114 static RAM on sockets
- Enhanced software

- Standard interfaces include: Audio Cassette Recorder interface with Remote Control; Full Duplex 20mA Teletype Interface; System Expansion Bus Interface; TV Controller Board interface; CRT Compatible interface

SALE ~~$225~~ SANYO MONITOR



VM4209 9" quality monitor **$149**

New Acoustic Coupler

$159

NCE Compumart Inc. breaks the price barrier with a new 300 baud RS 232C acoustic coupler. For only $159 you get an assembled tested stand alone unit which accepts the standard telephone handset for full or half duplex operation. Complete with power supply and carrier detect light, the unit is housed in an attractive case for desk top use.

TI PROGRAMMER

SALE!

Hexadecimal octal decimal calculator/converter

Reg. $59.95 NOW **$49.95**

A Home Computer & A Video Game in a Single Unit!

INTERACT Model One

- Color graphics on your TV
- Music and sound synthesizer
- Built-in cassette tape drive
- Large 53-key keyboard
- Two joysticks included
- 14 Free cassettes worth $294
- Edu-BASIC cassette included
- 8080 Microprocessor
- 8K RAM, 2K ROM

IN STOCK NOW! **$499**

If you can't decide between a sophisticated video game and a home computer than the Interact Model One is for you. Hook it to your color TV and you've got both! You can play one of the games which are included free or you can load Edu-BASIC and write your own games. You can save programs or data using the built-in cassette tape drive. Best of all you get over $294 worth of free cassettes including:

Edu-BASIC — Write programs
Concentration — The memory game
Dogfight — Aerial battles
Add-Em Up — A math game
Biorhythm — Plot your own
Blackjack — Wager video bucks
Compute-A-Color — Easy graphics
Hangman — The word game
Knockdown — A dice game
Maze — Computer maze
Regatta — Race your ship
Showdown — Gunfight
Trailblazer — Cut off your opponent
Blank — Save programs here

Rockwell AIM65

- Fully assembled tested and warranteed
- Addressing to 65K bytes (1K RAM on board)
- Includes onboard:
  - Powerful 4K ROM Debug Monitor
  - Sockets for additional 4K RAM 15K ROM
  - 20 Col ASCII Thermal printer 90 lines min
  - Full size (54 key) Alphanumeric keyboard
  - 20 char ASCII Alphanumeric display
  - Application Conn for TTY, 2 cassettes, 2 8 bit I/O ports
  - Expansion Conn (try our KIMSI S-100 board)
  - Both connectors totally KIM I compatible
- Easy Power connections $375

Options: 4K ROM Assembler Text Editor; 8K Basic ROM

10 DAY RETURN PRIVILEGE

- 90 day limited warranty
- Newest, most exciting products in personal computing
- Outstanding values
- All items fully assembled and tested unless otherwise stated.
- Send for our catalog for more detailed product information.

1-800 521 1534

• Michigan residents add 4% sales tax. • Foreign orders (except Canada) add 10% additional handling (U.S. currency only). • P.O.'s accepted from D&B rated companies. • Toll free phones open from 9:00 a.m.-7:00 p.m. EST Monday - Friday, 10:00 a.m.-5 p.m. Saturdays. • Open accounts invited — call for credit application. • Most items in stock for immediate shipment — call for delivery quotation. • Sorry — NO C.O.D.'s. • All prices subject to change without notice. • Write for our FREE catalog. • In the Ann Arbor area? Retail store open 11:00 a.m.-7:00 p.m. Tuesday-Friday, 10:00 a.m.-5:00 p.m. Saturdays.

MICRO

SUBSCRIPTION AND RENEWAL INFORMATION

If you are a subscriber to MICRO, then the code following your name on the mailing label is the number of the last issue your current subscription covers. If your code is 07, then this is your last issue. MICRO will NOT send out renewal notices. So, if your number is coming up, get your subscription renewal in soon. and, please check your label for correct address and notify us of any corrections or changes.

MICRO has been published bi-montly since 1977. Starting with the February 1979 issue, it will be published monthly. The single copy price is $1.50. Subscriptions are $12.00 per year in US.

Annual subscriptions in other countries are as listed below. All payments in US $.

Surface: Canada/Mexico	$15.00
All Other Countries	$18.00
Air Mail: Europe	$30.00
South America	$30.00
Central America	$25.00
All Other Countries	$35.00

Name: . . . .

Addr: . . . .

City: . . . .

State: . . . . Zip: . . . .

Country: . . . .

Amount: $ . . . . Start MICRO #: . . .

Send payment to:

MICRO, P.O. Box 3, S. Chelmsford, MA 01824, USA

Your name and address will be made available to legitimate dealers, suppliers, and other 6502 interests so that you may be kept informed of new products, current developments, and so forth - unless you specify that you do not wish your name released to these outside sources.

CONTINUOUS MOTION GRAPHICS OR HOW TO FAKE A JOYSTICK WITH THE PET

Alan K. Christensen
1303 Suffolk
Austin, TX 78723

When using the PET graphics to represent motion it becomes apparent that the BASIC supported routines are not fast enough to allow smooth movement. If the keyboard and screen are accessed directly the appearance of controlled motion can be greatly enhanced. As an example I will use a short game written in BASIC although the techniques can be used by machine language programs with even better results.

Let me first describe the game and then explain how the effects are produced. The initial appearance of the screen is two walls at the right and left sides of the screen with a ball and pound sign (#) which I will refer to as a bat (see figure 1). The ball goes into motion and appears to bounce off the top and bottom of the screen and the walls. Each time the ball strikes a wall it causes part of the wall to disappear. The ball will also bounce off the bat and the player is able to control the motion of the bat. This is done with the keys surrounding the number 5. As each key is pressed the bat moves in the same relative direction as that key was to key number 5 (see figure 2). For example if the number 8 is pressed the bat moves straight up. If the number 1 is pressed the bat moves along a diagonal towards the lower left side. The bat will continue to move for as long as the key is pressed. The object of the game is to make the ball strike the grey area of the left wall before it strikes the grey area of the right wall.

Lines 5-100 of the program are initialization. A special input array is set up (more about this later) and boundary conditions are set. Lines 80-90 print the walls. If the walls were placed directly on the screen the right wall could be one column further right and both walls could be extended one line. For this example I chose the simplest method of initializing the screen.

The boundaries are memory locations 32768 thru 33727. The characters on the PET screen are related directly to the values in memory locations 32768 thru 33767. The screen fills from left to right and is 40 characters wide therefore poking a value into byte 32768 causes a character to appear in the upper leftmost (home) position, byte 32768 + 39 is the upper rightmost postion, byte 32768 + 40 is the leftmost position of the second line and so forth until byte 33767 which is the lower rightmost character position. Table 1 gives the values for each character to cause it to appear on the screen. Lines 25 & 30 set the conditions to keep the ball and bat from moving off the top or bottom of the screen. The grey areas of the walls provide the boundaries for the sides of the screen. The right grey area is actually the reverse field (rvs) of the left grey area therefore a peek (32768) would return a value of key & = 38 + 64 (for shift) = 102 while a peek (32768 + 39) would return 102 + 128 (for rvs) = 230. This provides an easy method of detecting when the sides of the screen are reached (and in this example an indication that the game is over).

To provide motion for the ball a horizontal and vertical displacement are used. This is so the ball can move in directions other than up, down, sideways, or diagonal. X0 is 32768 + the column and Y0 is the line number with 0 as top line. X and Y are increments which are added to X0 and Y0 to get the next position. (P1 is the next position while P2 is the current position). If the next position is beyond the top or bottom of the screen the direction of Y is reversed and the next position is set to the current position (lines 120-125) this provides a bounce. The character on the screen at the next position is now checked (line 155). If this is equal to 35, the pound sign, (line 160) then the bat has struck the ball and it bounces off at a new angle. The magnitude of vector (S,Y) is fixed at 1 so that the ball cannot outrun the bat. If the next position has a screen value of 160 (32+128 for rvs blank) the white area of a wall was struck and the horizontal direction is reversed (line 180) but the new position is allowed to stand causing the ball to move into the wall. Lines 185-190 check for the winning or losing conditions. Finally in line 195 the next position is poked to the screen and the current position is blanked out (line 210). The current position is reset to the new position after looping to line 105 and the ball continues to move.

The bat is supposed to respond to the player and so a different movement scheme is used. The keyboard input routines supported by BASIC require one or more keys to be pressed and released for each input value to be received. This requires the player to tap at the keys like a woodpecker to control motion. To avoid this problem the program accesses byte 547 of the operating system working storage. When the interpreter is running the operating system places a unique value in this byte for each key that is pressed. (table 1 also gives these values, they are not the same as the screen character values). These values are then translated to a displacement for the bat.

The bat position is initialized and always kept at the actual address of the memory location which corresponds to the bats screen character position. A1 contains the next position while A2 contains the current position. In lines 35-45 an array E was set up with displacements stored at index values matching the values which may appear when any of the 8 keys surrounding number 5 is pressed. All other values of E are zero. By using the value at Peek (547) as an index to E the proper displacement for that key is obtained. For example when key number 2 is pressed, the value 18 appears at byte 547 and E(18)=40 which when added to the current position gives a next position one line lower(see lines 130-135) but if no key is pressed byte 547 contains 255 and since E(255)=0 the next position is the same as the current position and no motion takes place. The position is checked against the boundaries (line 140-150) and the screen is updated (lines 200-205). The program is now fast enough for the motion to appear continuous.

One drawback to this input scheme is that even though the keyboard buffer is not used to control the bat, it still fills up. Lines 310 and 320 show how the buffer had to be emptied before using the BASIC input routines again in line 370. When using the continuous keyboard input from a machine language routine it is important to leave the interrupt set to keyboard input or byte 547 may not get updated.

KEY	SCREEN VALUE	KEYBOARD VAL (547)	KEY	SCREEN VALUE	KEYBOARD VAL (547)
[	27	7	;	59	28
\	28	69	<	60	5
]	29	14	=	61	1
↑	30	59	>	62	12
←	31	75	?	63	20

The screen character values for a shift-key is the value of the key + 64. To get a reverse field (rvs) of a character (including shift-key characters) take the character value +128.

<u>Additional keyboard values:</u>

Home	74	
RVS	8	
STOP	4	(note pressing this key will still stop the program)
Up, down curser	66	
Sideways curser	73	
Del	65	

PROGRAM LISTING

```
5 REM ** WALL BREAK **
10 REM ALAN K. CHRISTENSEN
15 REM    AUSTIN, TEXAS
20 DIM E(256)
25 T = 32768
30 B = 33727
35 E(58) = -41 : E(50) = -40 : F(57) = -39
40 E(42) = - 1 : E(41) =  1
45 E(26) =  39 : E(18) =  40 : F(25) =  41
50 X0 = 32788
55 Y0 = 11
60 A1 = 33149
65 P1 = 33188
70 X = RND(1) -.5 : Y = SQR(1-X*X)
75 ?"↑"
     clr
80 FOR I = 1 TO 25
85 ?"▒↑^^"SPC(33)"^^▒"
     rvs
90 NEXT I
100 REM ** END OF INITIALIZATION **
105 A2 = A1 : P2 = P1
110 X0 = X0 + X : Y0 = Y0 + Y
115 P1 = X0 + 40 * INT(Y0)
120 IF P1 > B THEN Y = -Y : P1=P2
125 IF P1 < T THEN Y = -Y : P1=P2
130 I% = PEEK (547)
135 A1 = A1 + E(I%)
140 IF PEEK(A1) > 100 THEN A1=A2
145 IF A1 > B THEN A1=A2
150 IF A1 < T THEN A1=A2
155 P% = PEEK(P1)
160 IF P% <> 35 THEN 180
165 X = SGN(-X) * RND(1)
170 Y = SQR(1-X*X) * SGN(P2-A2)
175 P1 = P2
180 IF P%=160  THEN X=-X
185 IF P% = 102 THEN 300
190 IF P% = 230 THEN 400
195 POKE P1,87
200 POKE A1,35
205 IF A1<>A2 THEN POKE A2,32
210 IF P1<>P2 THEN POKE P2,32
215 GOTO 105
```

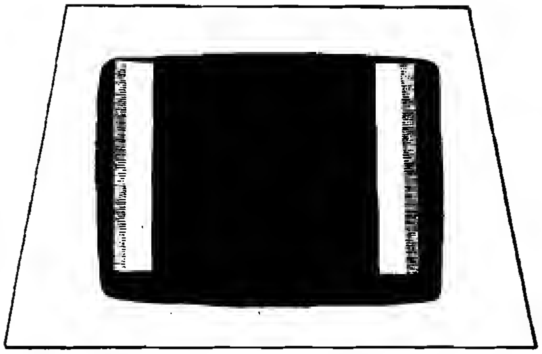


Figure 1
Showing the placement of the wall boundaries at the beginning of the game

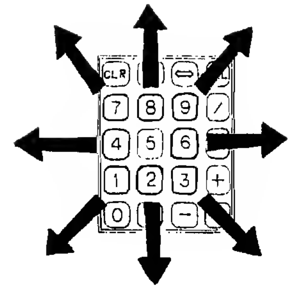


Figure 2

TABLE 1

KEY	SCREEN VALUE	KEYBOARD VAL (547)	KEY	SCREEN VALUE	KEYBOARD VAL (547)
@	0	15	blank	32	6
A	1	48	!	33	80
B	2	30	"	34	72
C	3	31	#	35	79
D	4	47	$	36	71
E	5	63	%	37	78
F	6	39	&	38	77
G	7	46	single quote '	39	70
H	8	38	(	40	76
I	9	53	)	41	68
J	10	45	*	42	33
K	11	37	+	43	17
L	12	44	comma ,	44	21
M	13	29	-	45	9
N	14	22	period .	46	2
O	15	60	/	47	49
P	16	52	0	48	10
Q	17	64	1	49	26
R	18	55	2	50	18
S	19	40	3	51	25
T	20	62	4	52	42
U	21	61	5	53	34
V	22	23	6	54	41
W	23	56	7	55	58
X	24	24	8	56	50
Y	25	54	9	57	57
Z	26	32	:	58	36

```
300 REM *** WINNER ***
310 GET A$
320 IF A$ <> "" THEN 310
330 ?"↑"SPC(12)"↑CONGRATULATIONS"
     home       rvs
340 FOR I = 1 TO 100 : NEXT I
350 ?"↑"SPC(12)"CONGRATULATIONS"
     home
360 FOR I = 1 TO 100 : NEXT I
370 GET A$
380 IF A$ = "" THEN 330
390 GOTO 50
400 REM *** LOSER ***
410 GET A$
420 IF A$ <> "" THEN 410
430 ?"↑"SPC(12)"↑SORRY↑^↑TRY↑^↑AGAIN"
     home       rvs  off  rvs off  rvs
440 FOR I = 1 TO 100 : NEXT I
450 ?"↑"SPC(12)"SORRY TRY AGAIN"
     home
460 FOR I = 1 TO 100 : NEXT I
470 GET A$
480 IF A$ = "" THEN 430
490 GO TO 50
500 END
```

STORAGE SCOPE REVISITED

Joseph L. Powlette
Donald C. Jeffery
Hall of Science
Moravian College
Bethlehem, PA 18018

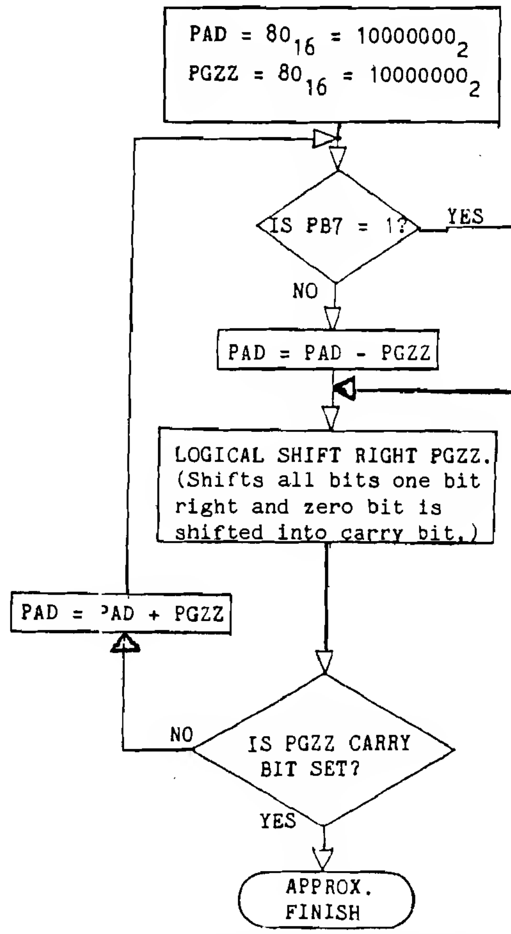
Marvin DeJong has written an excellent article (MICRO, No.2, pp.11-15, Dec 77-Jan 78) which serves to transform an ordinary oscilloscope into a storage scope. We have constructed several units for use in our laboratory and found them to be very useful. However, we would like to suggest a simple hardware change which will improve the quality of the circuits performance. Figure 1 is a photograph of the storage scope response to a triangular wave (14Hz and voltage offset) using DeJong's circuit. The cause of the irregularities seen in this figure was traced to the second OP-AMP which is used as a comparator. The slew rate of the CA3140 is not high enough to adequately accommodate the successive approximation software routine. Figure 2 shows the collection of data for the same wave with the second OP-AMP changed to a 531 high slew rate OP-AMP. The 531, which is readily available, has the same pin-out (in the TO-5 package) as the CA3140 but pin 4 must be connected to -15 volts rather than ground potential. Also, do not use a frequency compensation capacitor with the 531 since this will only decrease the slew rate of this OP-AMP in the comparator configuration. The 531 is not a FET input type and does not have the high input impedance (1.5 T) of the CA3140. If such a high impedance is desirable, one can use a CA3140 in the following configuration preceding the 531 non-inverting voltage input.

One should also note that:

1. There is a 7 bit version of the 1408 DAC. Specify 1408L8 for the 8 bit converter.
2. +5 volts should be connected to pin 13 of the 1408 (see MICRO, No. 6, p. 4, Aug-Sept, 1978)
3. The flow chart for the successive approximation routine is not correct.

DeJong is to be commended for this storage scope application. In fact, the performance of the program (with the above hardware change) approaches that of commercial units.

Flow Chart for
Successive Approximation
Analog to Digital Conversion



Correction to Successive Approximation - Micro, No.2, P. 13 Dec. 77 - Jan. 78

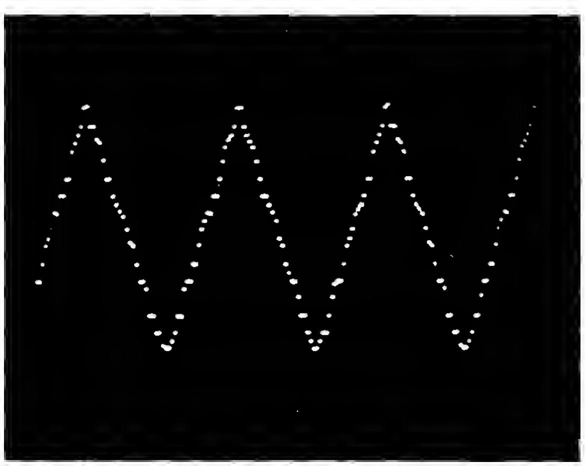


Figure 1

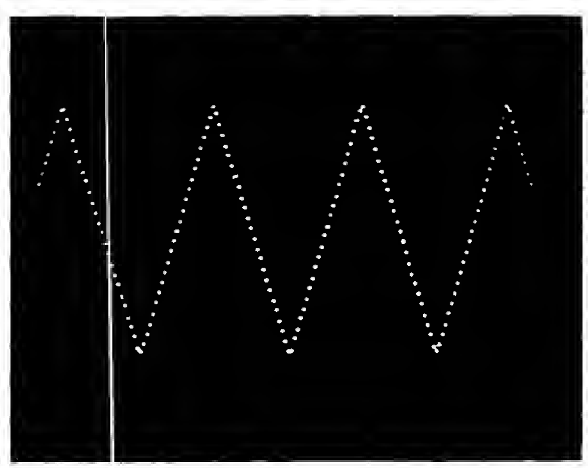
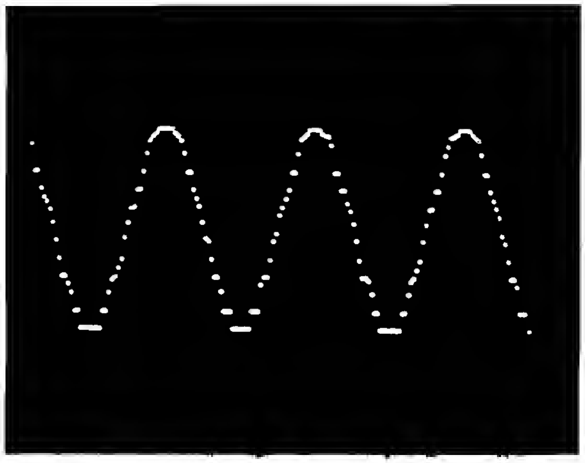
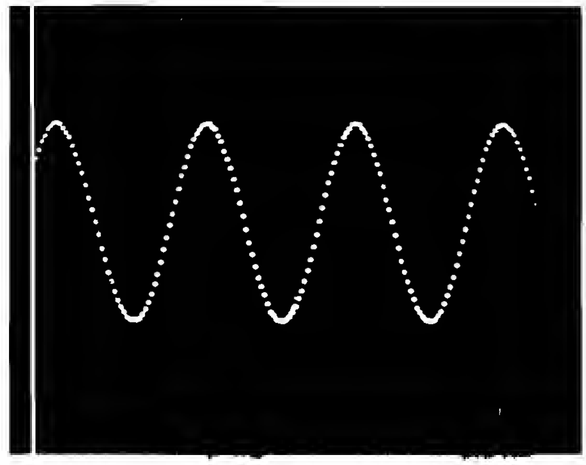


Figure 2



14 Hz Sine Wave
(Voltage Offset)

De Jong's Circuit



14 Hz Sine Wave
(Voltage Offset)

Modified Circuit

AN APPLE II PROGRAM RELOCATOR

Rick Auricchio
59 Plymouth Avenue
Maplewood, NJ 07040

After writing an Assembly-language program, the occasion often arises when one wishes the program to run in a different area of memory than that for which it was originally assembled. Relocating a program requires changing all absolute references within the program, so that it will run elsewhere in memory...this process is tedious, time-consuming, and repetitive WORK.

ENTER THE ELECTRONIC BRAIN

Behold! We have before us an electronic marval which thrives on such repetitive work! After all, why not just write a program to relocate others? Read on......

HERE'S WHAT IT TAKES

When a Relocating Assembler creates object code one of the items built is a Relocation Dictionary. This is actually a table of pointers to the program instructions that have absolute addresses; it also contains some flags for use by a relocating loader so that the latter can adjust the address references during the loading process.

Unfortunately, we don't have such a luxury when relocating most programs...all we have is raw machine language to work with. Our relocator will have to scan the subject program and find all absolute references which need adjustment.

FUNCTIONAL DESCRIPTION of RELOC8

The RELOC8 program will use the Apple's SWEET-16 utility for all 16-bit data and address manipulation; use of SWEET-16 saves a lot of 6502 code at the expense of some speed loss. In order to decipher the 6502 instructions of the subject program, Apple's Disassembler is used. (The disassembler, by the way, turns out to be a rather nice utility for things like this). In order to minimize user intervention, it was decided that RELOC8 would be run as part of a standard Apple Memory-Move command. After loading the subject program in its "old" memory location, one enters an Apple Move command to copy it to the "new" memory location, followed by Control-Y (which starts RELOC8 after the Move completes).

All absolute address references which lie within the range of the subject program will be updated. References to addresses outside the subject program (e.g. for Monitor calls) need not be changed.

USING RELOC8

To relocate a machine-language program, the following procedure is followed: load RELOC8 into the Apple and load the subject program into its "old" location. Type an Apple Move command to move the subject program to its "new" address followed by a space and control-Y. The RELOC8 program will print all modified instructions and then exit when it's done. For example, to relocate a subject program from "old" location 1500-1800, to "new" location 2A00-2D00, one would type the following command:

```
* 2A00<1500.1800M Yc
```

This is a standard "move" command, moving the program with the Apple Monitor; however, we follow the "M" with a space and a control-Y so that RELOC8 will be entered immediately following the move command. When it is entered, RELOC8 picks up the address values from the "move" command.

A FEW WORDS OF WARNING

There is something to watch out for while using RELOC8. Since it scans the subject program for absolute addresses, any data imbedded within the program may cause RELOC8 to think the data is an instruction. In that case, the data will be modified and RELOC8's opcode scan might get "out of sync" with the real instructions in the subject program. It's best to try and keep data separate from instructions; if RELOC8 does modify some data, you'll have to fix it before running the relocated program.

```
*****************************
*                           *
*  MACHINE-LANGUAGE         *
*   PROGRAM RELOCATOR       *
*                           *
*      -- RELOC8 --         *
*                           *
*  RICK AURICCHIO 10/26/78  *
*                           *
*   FOR THE APPLE-II        *
*                           *
*****************************
*
* --- SWEET-16 REGISTERS
*
AC        EQU      0     R0:ACCUMULATOR
OB        EQU      1     R1:OLD BASE
OE        EQU      2     R2:OLD END
NB        EQU      3     R3:NEW BASE
NE        EQU      4     R4:NEW END
RB        EQU      5     R5:RELOCATION BIAS
*
```

```
00000000          ACL       EQU     0
00000001          ACH       EQU     1
00000002          OBL       EQU     2
00000003          OBH       EQU     3
00000004          OEL       EQU     4
00000005          OEH       EQU     5
00000006          NBL       EQU     6
00000007          NBH       EQU     7
00000008          NEL       EQU     8
00000009          NEH       EQU     9
                  *
                  *
0000F689          SWEET16   EQU     X'F689'      SWEET-16 INTERPRETER
0000F88E          INSDS2    EQU     X'F88E'      DISASSEMBLE WITHOUT PRINT
0000F8D0          INSTDSP   EQU     X'F8D0'      DISASSEMBLE SINGLE INSTR.
                  *
0000002F          LENGTH    EQU     X'2F'        DISASSEMBLED INSTR LENGTH
0000003C          A1L       EQU     X'3C'        WORK BYTES FOR MONITOR
0000003D          A1H       EQU     X'3D'
00000040          A3L       EQU     X'40'
00000041          A3H       EQU     X'41'
00000044          A5L       EQU     X'44'
00000045          A5H       EQU     X'45'
0000003A          PCL       EQU     X'3A'        PC LOW FOR DISASSEMBLER
0000003B          PCH       EQU     X'3B'        ..TAKE A GUESS...


                  * ENTRY IS VIA CONTROL-Y AFTER
                  *  MOVING PROGRAM TO ITS NEW
                  *  LOCATION IN MEMORY. THE
                  *  VALUES FROM THE APPLE 'MOVE'
                  *  COMMAND WILL BE PRESENT IN
                  *  THE MONITOR WORK AREAS UPON
                  *  ENTRY TO RELOC8.
                  *
0300                        ORG     X'0300'      ORG TO PAGE 3
0300     A5 40    RELOC8    LDAZ    A3L          MOVE OLD BASE
0302     85 02              STAZ    OBL
0304     A5 41              LDAZ    A3H
0306     85 03              STAZ    OBH
                  *
0308     A5 3C              LDAZ    A1L          MOVE OLD END (+1)
030A     85 04              STAZ    OEL
030C     A5 3D              LDAZ    A1H
030E     85 05              STAZ    OEH
                  *
0310     A5 44              LDAZ    A5L          MOVE NEW BASE
0312     85 06              STAZ    NBL
0314     A5 45              LDAZ    A5H
0316     85 07              STAZ    NBH
```

```
                         *
                         * --- COMPUTE NEW END AND
                         *      RELOCATION BIAS.
                         *
0318      20 89 F6               JSR     SWEET16       GO TO SWEETIE
031B      23                     LD      NB
031C      B1                     SUB     OB            RELOCATION BIAS
031D      35                     ST      RB             IS DIFFEREOCE
031E      22                     LD      OE
031F      B1                     SUB     OB            COMPUTE SIZE
0320      A3                     ADD     NB            ADD TO NEW BASE
0321      34                     ST      NE             AND WE HAVE NEW END
0322      00                     RTN                   6502 MODE!


                         *
                         * SCAN THE PROGRAM FOR A 3-BYTE
                         *  INSTRUCTION. ANY OTHERS DON'T
                         *  HAVE TO BE RELOCATED. IF THE
                         *  ADDRESS IS OUTSIDE THE PROGRAM,
                         *  THEN WE CAN LEAVE IT ALONE.
                         *  OTHERWISE, UPDATE IT BY ADDING
                         *  THE RELOCATION BIAS.
                         *
0323      A0 00          GETINST LDYIM   0             DUMMY INDEX
0325      B1 06                  LDAIY   NBL           GET OPCODE
0327      20 8E F8               JSR     INSDS2        GET ITS LENGTH
032A      A5 2F                  LDAZ    LENGTH        CHECK LENGTH
032C      C9 02                  CMPIM   2             3 BYTES?
032E      D0 24                  BNE     NXTINST       =>NOPE. SKIP IT.
                         *
                         * IF THE ADDRESS IS WITHIN THE
                         *  PROGRAM, RELOCATE IT.
                         *
0330      20 89 F6               JSR     SWEET16       HI, SWEETIE!
0333      E3                     INR     NB            BUMP TO ADDRESS
0334      63                     LDD     NB            GET BOTH BYTES
0335      D1                     CPR     OB            >= OLD BASE?
0336      02 2A                  BNC     NXT1          =>LOWER. NO CHANGE.
0338      D2                     CPR     OE            <= OLD END?
0339      03 27                  BC      NXT1          =>HIGHER. NO CHANGE.
                         *
                         * ADD RELOCATION BIAS.
                         *
033B      A5                     ADD     RB            ADD BIAS
033C      F3                     DCR     NB            BACK UP TO
033D      F3                     DCR     NB             ADDRESS AGAIN
033E      73                     STD     NB            STUFF BACK THERE
                         *
                         * --- ANNOUNCE THE CHANGE --- *
                         *
033F      23                     LD      NB            BACK UP POINTER
0340      F0                     DCR     AC             TO OPCODE
0341      F0                     DCR                     FOR THE
0342      F0                     DCR                      DISASSEMBLER
0343      00                     RTN                   BACK TO 6502 MODE
0344      A5 00                  LDAZ    ACL           MOVE POINTER
0346      85 3A                  STAZ    PCL            TO PCH/PCL
0348      A5 01                  LDAZ    ACH             FOR THE
034A      85 3B                  STAZ    PCH              DISASSEMBLER
034C      20 D0 F8               JSR     INSTDSP       PRINT MODIFIED INSTR.
034F      20 89 F6               JSR     SWEET16       RE-ENTER SWEET16 TO
0352      01 0E                  BR      NXT1           CONTINUE...
```

```
                    *
                    * WE'VE GOT A 1 OR 2 BYTE
                    *  INSTRUCTION. UPDATE THE
                    *  NB POINTER TO THE NEXT
                    *  INSTRUCTION.
                    *
0354      18        NXTINST  CLC
0355      69 01              ADCIM    1            UPDATE LENGTH: 1/2/3
0357      85 00              STAZ     ACL          GET LENGTH
0359      A9 00              LDAIM    0            HI=0
035B      85 01              STAZ     ACH
035D      20 89 F6           JSR      SWEET16      BACK TO SWEET16
0360      A3                 ADD      NB           BUMP IT
0361      33                 ST       NB           PUT BACK THERE
                    *
                    * CHECK TO SEE IF WE'RE DONE
                    *  WITH THE PROGRAM YET.
                    *
0362      23        NXT1     LD       NB           GET CURRENT ADDRESS
0363      D4                 CPR      NE           OVER THE END?
0364      03 04              BC       DONE         =>YUP. ALL DONE!
0366      00                 RTN                   =>NO. BACK TO THE
0367      B8                 CLV                      6502 MODE FOR
0368      50 B9              BVC      GETINST          MORE WORK!
                    *
                    * ALL DONE. EXIT TO MONITOR.
                    *
036A      00        DONE     RTN                   6502 MODE, PLEASE!
036B      60                 RTS                   BACK TO MONITOR!
                    *
                    *
03F8                         ORG      X'03F8'      CONTROL-Y ENTRY
03F8      4C 00 03           JMP      RELOC8       ROLL STONE, GATHER MOSS...
                    *
                             END
```

SYM-1 TAPE DIRECTORY

John Gieryic
2041 138th Avenue N.W.
Andover, MN 55303

The SYM-1's high speed tape format enables recording and loading of 1K of RAM in just a few seconds (185 bytes per second). This quick and easy means of saving and restoring memory will have you SYM-1 owners quickly wrapped up in tape. With the possibility of 254 ID's (01 thru FE) you may forget which ID's you've already used or where you stored a particular identifier. Maintaining records sometimes seems secondary when you are eagerly pursuing an idea.

This program will refresh your memory quickly. When DIRECTORY "finds" a tape record it will extract the ID, startind address and ending address + 1. This information will be paraded across the LED's in much the same format used when you saved the data on tape. The program will then continue its search for more records. The process is terminated by pressing the RST key.

The first part of the program (locations 205 thru 232) is taken from the monitor routine LOADT. Since this is not a subroutine (callable by a JSR), I had to copy the necessary logic into my program. The last part of the program makes extensive use of subroutine calls to two of my own subroutines and several of the monitor's. Any newcomers to programming should take time to trace through this in order to see the power of subroutines.

SYM TAPE DIRECTORY

High Speed Format Only: START: GO 200 CR

TAPE FORMAT:

256 Sync Char * ID SAL SAH EAL+1 EAH+1

DATA / CKL CKH EOT EOT

This program will extract the tape identifier (ID), the starting address (SAL and SAH), the ending address (EAL and EAH) and will "parade" this information on the LED's. The program will then go back to the tape and search for the next record. The program is terminated by pressing the RST key.

```
                          SYM TAPE DIRECTORY

                          SYM REFERENCES

                          ACCESS *     $8B86
                          START  *     $8DB6
                          SYNC   *     $8D82
                          RDCHTX *     $8DDE
                          RDBYTX *     $8E28
                          RDBYTH *     $8DE2
                          OUTDSP *     $89C1
                          NIBASC *     $8309
                          SCAND  *     $890B
                          DISBUF *     $A641

                          DDRIN  *     $A002
                          VIAACR *     $A00B
                          LATCHL *     $A004
                          MODE   *     $00FD

                                 ORG   $0000

0000  00                  ID     =     $00    TAPE ID LOCATION
0001  00                  SAL    =     $00
0002  00                  SAH    =     $00
0003  00                  EAL    =     $00
0004  00                  EAH    =     $00
0005  00                  TEMP   =     $00
0006  00                  LCNT   =     $00    LOW LOOP COUNTER
0007  00                  HCNT   =     $00    HIGH LOOP COUNT
```

```
0200                    ORG   $0200  PROGRAM ORIGIN

0200 20 86 8B  BEGIN    JSR   ACCESS ENABLE SYM PROTECTED MEMORY
0203 A0 80              LDYIM $80    SET HIGH SPEED MODE
0205 20 B6 8D           JSR   START  INIT TAPE ROUTINES
0208 AD 02 A0           LDA   DDRIN
020B 29 BF              ANDIM $BF
020D 8D 02 A0           STA   DDRIN
0210 A9 00              LDAIM $00
0212 8D 0B A0           STA   VIAACR
0215 A9 1F              LDAIM $1F    SET UP TIMER
0217 8D 04 A0           STA   LATCHL
021A 20 82 8D  FIND     JSR   SYNC   SEARCH TAPE FOR RECORD
021D 20 DE 8D  READ     JSR   RDCHTX GET CHARACTER
0220 C9 2A              CMPIM '*     COMPARE FOR ASTERISK
0222 F0 06              BEQ   TEST   MATCH
0224 C9 16              CMPIM $16    TEST SYNC CHAR
0226 D0 F2              BNE   FIND
0228 F0 F3              BEQ   READ

022A A5 FD     TEST     LDA   MODE
022C 29 BF              ANDIM $BF
022E 85 FD              STA   MODE
0230 20 28 8E           JSR   RDBYTX GET ID
0233 85 00              STA   ID     SAVE ID
0235 20 28 8E           JSR   RDBYTX GET SAL FROM TAPE
0238 85 01              STA   SAL    SAVE
023A 20 28 8E           JSR   RDBYTX GET SAH FROM TAPE
023D 85 02              STA   SAH    SAVE
023F 20 E2 8D           JSR   RDBYTH GET EAL
0242 85 03              STA   EAL    SAVE
0244 20 E2 8D           JSR   RDBYTH GET EAH
0247 85 04              STA   EAH    SAVE
0249 A9 00              LDAIM $00    CLEAR OUT DISPLAY BUFFER
024B 8D 41 A6           STA   DISBUF
024E 8D 42 A6           STA   DISBUF +01
0251 8D 43 A6           STA   DISBUF +02
0254 8D 44 A6           STA   DISBUF +03
0257 8D 45 A6           STA   DISBUF +04
025A A5 00              LDA   ID     TAPE ID
025C 20 96 02           JSR   DISPL  SEND IT TO DISPLAY
025F A9 2D              LDAIM '-     ASCII DASH
0261 20 C1 89           JSR   OUTDSP SEND IT TO DISPLAY
0264 20 B5 02           JSR   DELAY  PAUSE
0267 A5 02              LDA   SAH    START ADDRESS HIGH
0269 20 96 02           JSR   DISPL  SEND TO DISPLAY
026C A5 01              LDA   SAL    START ADDRESS LOW
026E 20 96 02           JSR   DISPL  SEND TO DISPLAY
0271 A9 2D              LDAIM '-     DASH
0273 20 C1 89           JSR   OUTDSP DISPLAY IT
0276 20 B5 02           JSR   DELAY  PAUSE
0279 A5 04              LDA   EAH    END ADDRESS HIGH
027B 20 96 02           JSR   DISPL
027E A5 03              LDA   EAL    END ADDRESS LOW
0280 20 96 02           JSR   DISPL
0283 A9 00              LDAIM $00    ADD 2 TRAILING BLANKS
0285 20 C1 89           JSR   OUTDSP
0288 20 B5 02           JSR   DELAY
028B A9 00              LDAIM $00
028D 20 C1 89           JSR   OUTDSP
0290 20 B5 02           JSR   DELAY
0293 4C 00 02           JMP   BEGIN  GO TO NEXT RECORD ON TAPE
```

```
                          SUBROUTINE DISPL

                          ENTRY  LDA (BINARY DATA)
                                 JSR DISPL

                          THE UPPER FOUR BITS IN THE A REGISTER ARE
                          CONVERTED TO THEIR ASCII EQUIVALENT, SENT
                          TO THE DISPLAY VIA SUBROUTINE DELAY.  NEXT
                          THE PROCESS IS REPEATED WITH THE LOWER
                          FOUR BITS.

0296 85 05                DISPL  STA    TEMP    SAVE A REGISTER
0298 6A                          RORA           RIGHT JUSTIFY LEFT FOUR BITS
0299 6A                          RORA
029A 6A                          RORA
029B 6A                          RORA
029C 29 0F                       ANDIM  $0F     MASK TO FOUR BITS
029E 20 09 83                    JSR    NIBASC  CONVERT TO ASCII
02A1 20 C1 89                    JSR    OUTDSP  SEND TO DISPLAY
02A4 20 B5 02                    JSR    DELAY   PAUSE
02A7 A5 05                       LDA    TEMP    RESTORE A
02A9 29 0F                       ANDIM  $0F     MASK OFF TO LOWER FOUR BITS
02AB 20 09 83                    JSR    NIBASC  CONVERT TO ASCII
02AE 20 C1 89                    JSR    OUTDSP  SEND TO DISPLAY
02B1 20 B5 02                    JSR    DELAY   PAUSE
02B4 60                          RTS            RETURN

                          SUBROUTINE DELAY

                          ENTRY  JSR DELAY

                          THIS ROUTINE WILL CALL SCAND FOR A PERIOD
                          OF TIME IN ORDER TO ILLUMINATE THE 6 LED'S

02B5 A9 00                DELAY  LDAIM  $00     INIT LOOP COUNTERS
02B7 85 06                       STA    LCNT
02B9 85 07                       STA    HCNT
02BB 20 0B 89             WAIT   JSR    SCAND   SYM DISPLAY
02BE E6 06                       INC    LCNT
02C0 D0 F9                       BNE    WAIT    DELAY
02C2 E6 07                       INC    HCNT
02C4 A5 07                       LDA    HCNT    TEST COUNTER
02C6 C9 03                       CMPIM  $03
02C8 D0 F1                       BNE    WAIT
02CA 60                          RTS
```


"THE BEST OF MICRO VOLUME 1"

Even though we had extra copies of MICRO printed we could not keep up with the demand for back issues. We have run out of all back issues and all copies of "All of MICRO Volume 1". Since a lot of people who are just finding out about MICRO or are just getting into the 6502 world still want the information which was contained in the first year of MICRO, we have decided to print "The BEST of MICRO Volume 1".

This will contain most of the articles but none of the advertising. A few articles which were topical and are now out-of-date will be dropped and all known microbes will be corrected back in the original articles. The book will be organized by subject. Aside from these minor changes, the content will be identical to that of MICRO numbers 1 through 6. If you already have them, you will not profit by getting the new edition. If you do not have them, then this will be the only way to get the information.

"The BEST of MICRO Volume 1" is available now. 176 pages plus five reference cards, soft cover. Price: $6.00 at your dealer or by surface mail anywhere in the world - $7.00.

Send your Check or Money Order to:

The BEST of MICRO
P.O. Box 3
So. Chelmsford, MA 01824

INSIDE PET BASIC

Jim Butterfield
14 Brooklyn Avenue
Toronto, Ontario
Canada M4M 2X5

PET BASIC is pretty good: fast, powerful, and flexible. Most of the time you can write programs without ever needing to know what's inside. But there are a few handy things that you can't do without "dissecting" BASIC. Let's tak a couple of examples. Suppose you want to look through a big program for some reason. You might have a small bug: say a variable, X4, ends up with a wrong value, and you want to find out why. You could list the program, a screenful at a time, looking for every time X4 is used; bu eye fatigue starts to set in. Wouldn't it be nice to have a utility program to do the scanning for you?

Program FIND

Program FIND will do the job for you. To write such a program, though, we need to know how BASIC is built. The first line of your BASIC program starts at address 1025 (or 0401 hexadecimal). That's where we must start our search. Each BASIC line will have the following format: The first two locations contain a pointer to the next line of BASIC; or if they contain zeros, there is no next line and this is the end of your program. The next two locations contain the BASIC line number. After that (starting at the fifth location) we have the BASIC line itself. It's mostly in ASCII code, but keywords such as FOR, PRINT, or SQR are stored as special codes known as "tokens". At the end of the line we'll find the value zero.

How do we use this information to scan BASIC for a given expression? First, we set our address, A, to 1025; that's where BASIC starts. Next, we skip over the first four bytes (pointer and line number) and search from A+4 to the end of the BASIC line. We'll recognize the end-of-line by the zero at the end. If we find the expression we want, we can output the line number by obtaining it from A+2 and A+3. It's in binary, so we use the expression 256*PEEK(A+3)+PEEK(A+2) - printing this value will print the line number.

When we reach the end of the BASIC line, we must go to the next line, of course. It will be right behind the zero that marked the end of our previous line; or we can use the pointer to jump ahead with A=256*PEEK(A+1)+PEEK(A). If the pointer is zero, we know that we have come to the end of the BASIC program and can stop.

Program RESEQUENCE

Let's move on to something more complicated. Suppose you want to renumber your BASIC program. Since we know how the line numbers are stored in BASIC, it seems easy; we'll just change them to the new values. There is a hitch, however. What happens if your program contains a GOTO 300 statement - and now line 300 is renumbered so that it becomes line 380? Problems - that's what happens.

What we must do is search out all the GOTOs and GOSUBs, including those included in ON.. statements, and be ready to change the old line numbers to new ones. One way of doing this is to build a table of "old" addresses, match them with the "new" line numbers, and then correct them after renumbering has been accomplished. To help make things more complicated, we have two different ways of using the THEN statement. If we have a line such as IF J=12 THEN Y=2, there is no line number reference to correct. On the other hand, if we have IF J=12 THEN 530, we must be ready to fix up 530, replacing it with a new line number if necessary.

More difficulties: if we have a statement which says, for example, GOTO 5, and with the renumbering we want to change it to GOTO 100, we won't have space! And making space isn't that easy: you may recall that the lines of BASIC are "chained" together with pointers; if we lengthen a BASIC line, all the pointers will need to be fixed up! This last problem is too tough to resolve in a simple manner - let's sidestep it by printing a warning notice if it should occur.

How do we approach this job? We separate the program into three phases. Phase 1 looks through the program for line number references and builds a table. Phase 2 does the actual renumbering (the easiest part of the whole job). Phase 3 looks through the program again and corrects the line number references. How do we look through the program? The same way as with program FIND. We're looking for three keywords: GOTO (token 137), GOSUB (141) and THEN (167). Sometimes we'll also allow a comma (44) so that statements such as ON X GOTO 100,200,300 will be allowed. You'll see this testing for tokens on line 60220 of RESEQUENCE.

If we find one of these keywords, we must convert the following ASCII numbers into a value V corresponding to the line number. During Phase 1, we build these line numbers into a table at 60090. Phase 2 is a snap. In lines 60030-60040 we change the line number and then check to see if the old number was in table V%. IF so, we fill in the cross-reference. Phase 3 is the long one. We must repeat the search of Phase 1. Then, in 60110 to 60150 we must build the new line number (in ASCII) and insert it - with appropriate tests and warning notices.

Making Them Work

Both FIND and RESEQUENCE are written in BASIC. That means that they will have to reside in PET's memory along with the programs they are dealing with. RESEQUENCE is constructed so that it doesn't renumber itself, of course; and FIND will examine itself, reporting any occurences of the search string. Another problem arises, however: how can you get two programs into the PET at the same time? We need to load either FIND or RESEQUENCE together with the program that is being processed. A normal PET load wipes out the old program when a new one is loaded. You could alsays add FIND or RESEQUENCE by entering it at the keyboard; this would add the utility program to the existing program in memory. But such a procedure is lengthy and it would be easy for errors to creep in. There must be a better way. One good way is to use the screen as a "holding buffer". You could load program FIND, and list it onto the screen. Then load the program you wand to search. FIND will be wiped out of memory, but it's still on the screen - so you

can move the cursor back to displayed line 9000, and hit RETURN eight times. FIND will be restored to memory, where it now shares space with the program to be scanned. This doesn't work too well with a longer program like RESEQUENCE, however. The program is too big to fit on the screen - much too big. There must be another even better way. Larry Tessler of Sphinx opened the door with his program UNLIST, which made true program merging possible for the first time. Since this breakthrough, an even better method has been devised by Brad Templeton of Toronto.

UNLIST - A Procedure for Merging Programs

Here's how it works. Be sure to follow the instructions carefully and exactly. Prepare the programs you will want to merge in the following manner. Load the program. Place a blank tape into your cassette unit. Now type:

```
OPEN 1,1,1:CMD 1:LIST
```

When the tape stopes, type:

```
PRINT#1:CLOSE1
```

and your merge tpae is ready. At a later time, when you want to merge the program, here's what to do. First, mount the merge tape you previously prepared and type OPEN 1. Now clear the screen, give exactly four cursor downs, and type the following, but DO NO HIT RETURN:

```
POKE611,1:POKE525,1:POKE527,13:?"h"
```

(h is cursor home; shows as reverse S). Don't hit return: press cursor home and give six (6) cursor downs. Now type exactly the same line (two lines below the first line) and then hit RETURN. The tape will more; the merge will take place; and finally, an error notice will print between the two lines. Stop the tape if it's still going, and then type CLOSE1. Miraculously the merge has taken place!

How does it work? It's a little complex; but if I hinted that POKE 611,1 thransfers control away from the PET's keyboard to the cassette tape, you'd have part of the story. And if I mentioned that poking 525 and 527 simulates a RETURN key being hit, you'd have another part. But, you don't need to know what makes it work in order to use it. Use it; benefit from it; and enjoy it.

the following, but DO NOT HIT RETURN:

FIND for PET

Need to search a program for an express, a variable, or a keyword? Slip program FIND in behind your program (it's not very long) - then insert a line 1 to say what to search for ... and the job's done. Every line in memory which contains the same expression as line 1 will be reported. This includes line 1 itself, of course, and any lines in program FIND ... as well as the program you're searching. The program is listed here spaced out for readability - close in the spaces when you input to save space.

```
9000 A=1025 : X=PEEK(1029) FOR J=1 TO 1E3 : FOR
     K=A+4 TO A+83
9001 P=PEEK(K) : IF P=X THEN GOSUB 9005
9002 IF P<>0 THEN NEXT K
9003 A=256*PEEK(A+1)+PEEK(A) : IF A>0 THEN
     NEXT J
9004 STOP
9005 FOR L=1 TO 80 : Y=PEEK(1029+L) : FI Y=0
     THEN ? 256*PEEK(A+3)+PEEK(A+2); : RETURN
9006 IF Y=PEEK(K+L) THEN NEXT L
9007 RETURN
```

Example: to find all FOR statements in a program; insert FIND (above) and then insert line 1

```
1 FOR
```

Now invoke FIND with RUN 9000. The program will print 1 followed by any program lines containing FOR followed by 9000 9000 9005 (9000 prints twice because it contains two FORs).

FOR is a keyword, and doesn't store as three separate characters, so you wouldn't find it if you searched for characters FO. This can be handy: if you were looking for variable F you wouldn't get all the FORs printed.

Modifications: if you squeezed P=0 just ahead of RETURN on line 9005 (it's a tight squeeze) a line number would print only once even when it had multiple matches; you might or might not want this feature.

IMPORTANT: Don't forget to wipe out line 1 and program FIND when you're finished with them.

RESEQUENCE for PET

```
60000 END
60010 TO= : DIM V%(100),W%(100) : GOSUB 60160 :
      FOR R=1 TO 1E3 : GOSUB 60210
60020 IF G THEN GOSUB 60090 : NEXT R
60030 GOSUB 60160 : FOR R=1 TO 1E3 : N=INT
      (M/256) : POKE A-1,M-N*256
60040 POKE A,N : V=L : GOSUB 60070 : W%(J)=M :
      GOSUB 60170 : IF G THEN NEXT R
60050 GOSUB 60160 : FOR R=1 TO 1E3 : GOSUB 60210
      : IF G THEN GOSUB 60110 : NEXT R
60060 ?"*END*" : END
60070 J=0 : IF T<>0 THEN FOR J=1 TO T : IF V%(J)
      <> V THEN NEXT J : J = 0
60080 RETURN
60090 IF V<>0 THEN GOSUB 60070 : IF J=0 THEN T=
      T+1 : V%(T)=V
60100 RETURN
60110 GOSUB 60070 : IF J=0 THEN RETURN
60120 W=W%(J) ; IF W=0 THEN ?"GO";"L";L;"?":
      RETURN
60130 FOR D=A TO B+1 STEP-1 : X=INT(W/10) :
      Y=W-10*X+48 : IF W=0 THEN Y=32
60140 POKE D,Y : W=X : NEXT D : IF W=0 THEN
      RETURN
60150 ?"INSERT";W%(J);"L";L : RETURN
60160 F=1025 : M=90
60170 A=F : M=M+10
60180 F=PEEK(A)+PEEK(A+1)*256 : L=PEEK(A+2)+
      PEEK(A+3)*256 : A=A+3 : G=L<6E4
60190 RETURN
60200 S=0
60210 V=0 : A=A+1 : B=A : C=PEEK(A) : IF C=0
      THEN GOSUB 60170 : ON G+2 GOTO 60210,60190
60220 IF C<>137 AND C<>141 AND C <>167 AND C<>S
      GOTO 60200
```

```
60230 A=A+1 : C=PEEK(A)-48 : IF C=-16 GOTO 60230
60240 IF C>=0 AND C<9 THEN V=V*10+C : GOTO 60230
60250 S+44 : A=A-1 : RETURN
```

RESEQUENCE can sit quietly behind your program. When you say RUN 60010, your program is renumbered. RESEQUENCE gives error notices if:

A. a GOTO or GOSUB statement wants to go to a non-existant line;
B. there isn't enough room for a new (higher) line number.

In both cases you're given the (new) line number where this happens. RESEQUENCE doesn't run fast (allow about a second per line, more for large programs), but it's dependable and very useful.

Program comments: Line 6000 stops the user program if it gets here. Lines 60010-60020 extract all GOTO, GOSUB, and THEN references and build them into a table. Lines 60030-60040 renumber all lines, and cross-references the table if needed. Line 60050 updates all line references.

Subroutines: 60070 looks for an entry in the line number table. 60090 inserts a new entry into the table. 60110 revises a line number reference. 60160 starts a new scan of the user program; 60170 continues the scan with the next line. 60210 scans the user program for GOTOs, etc.; value S is used to accomodate ON A GOTO ... type situations.

AN APPLE II PAGE 1 MAP

M.R. Connolly Jr.
5009 Rickwood Ct. NW
Huntsville, AL 35810

In the Apple II, the on-screen text is stored in locations $400 through $7FF. Trying to determine just where a particular spot resides in memory isn't easy. The page lines are stored neither consecutively nor sequentially. The APPLE page 1 map shows in hex and decimal the starting and ending locations of each line on the screen. Any given line is sequential from space 1 through space 40; eg, the 20th position of any line is equal to the beginning location +19 decimal or 14 hex.

The value of the page map becomes apparent when used with a listing of the interpretation of numbers stored in the map. Any normal, inverse, or flashing character, or white block, black block, or cursor block may be positioned merely by poking the correct value in the location storing the page position you require.

You might pass this off as just "nice to know" information, but it is very useful if, for instance, you are trying to make an impressive title page for a program you've spent weeks writing. Run the following short program, then try to duplicate it without using the page map and the character chart. It isn't easy!

```
10 CALL -936: FOR I = 1205 TO 1217: POKE I,32: POKE I+ 512,32: NEXT I
20 FOR I = 1333 TO 1589 STEP 128: POKE I,32: POKE I+ 12,32: NEXT I
30 POKE 1463,141: POKE 1465,9: POKE 1467,67: POKE 1469,18: POKE 1471,207
40 GOTO 40
```

MAP OF LINE AND SPACE LOCATIONS FOR TEXT PAGE 1, APPLE II COMPUTER

LINE	LOCATION HEX	LOCATION DECIMAL
1	400-427	1024-1063
2	480-4A7	1152-1191
3	500-527	1280-1319
4	580-5A7	1408-1447
5	600-627	1536-1575
6	680-6A7	1664-1703
7	700-727	1792-1831
8	780-7A7	1920-1959
9	428-44F	1064-1103
10	4A8-4CF	1192-1231
11	528-54F	1320-1359
12	5A8-5CF	1448-1487
13	628-64F	1576-1615
14	6A8-6CF	1704-1743
15	728-74F	1832-1871
16	7A8-7CF	1960-1999

17	450-477	1104-1143
18	4D0-4F7	1232-1271
19	550-577	1360-1399
20	5D0-5F7	1488-1527
21	650-677	1616-1655
22	6D0-6F7	1744-1783
23	750-777	1872-1911
24	7D0-7F7	2000-2039

Not used for on-screen display: 478-47F; 4F8-4FF; 578-57F; 5F8-5FF; 678-67F; 6F8-6FF; 778-77F; 7F8-7FF

MACHINE INTERPRETATION OF VALUES STORED IN $4ØØ.7FF APPLE II COMPUTER

FIGURE	NORMAL	INVERSE	FLASH	FIGURE	NORMAL	INVERSE	FLASH
@	128,192	0	64	!	161,225	33	97
A	129,193	1	65	"	162,226	34	98
B	130,194	2	66	#	163,227	35	99
C	131,195	3	67	$	164,228	36	100
D	132,196	4	68	%	165,229	37	101
E	133,197	5	69	&	166,230	38	102
F	134,198	6	70	'	167,231	39	103
G	135,199	7	71	(	168,232	40	104
H	136,200	8	72	)	169,233	41	105
I	137,201	9	73	*	170,234	42	106
J	138,202	10	74	+	171,235	43	107
K	139,203	11	75	,	172,236	44	108
L	140,204	12	76	-	173,237	45	109
M	141,205	13	77	.	174,238	46	110
N	142,206	14	78	/	175,239	47	111
O	143,207	15	79	Ø	176,240	48	112
P	144,208	16	80	1	177,241	49	113
Q	145,209	17	81	2	178,242	50	114
R	146,210	18	82	3	179,243	51	115
S	147,211	19	83	4	180,244	52	116
T	148,212	20	84	5	181,245	53	117
U	149,213	21	85	6	182,246	54	118
V	150,214	22	86	7	183,247	55	119
W	151,215	23	87	8	184,248	56	120
X	152,216	24	88	9	185,249	57	121
Y	153,217	25	89	:	186,250	58	122
Z	154,218	26	90	;	187,251	59	123
[	155,219	27	91	<	188,252	60	124
\	156,220	28	92	=	189,253	61	125
]	157,221	29	93	>	190,254	62	126
^	158,222	30	94	?	191,255	63	127
_	159,223	31	95				
(BLOCK)	160,224 □	32 ■	96 □⇔■				

6502 BIBLIOGRAPHY
PART VII

William R. Dial
438 Roslyn Avenue
Akron, OH 44320

380. Gordon, H. T. "Use of NOPCODES as Executable Labels", Dr Dobb's Journal 3, Issue 8 No 28 pg 29 (Sept., 1978). Discusses the use of nopcodes in 650X devices. Classifies these as monops (6 listed) and binopcodes (5 listed) and trinops (8 listed).

381. Swank, Joel "A Programmable IC Tester for KIM", DDJ 3 Issue 8 no 28 pg 33 (Sept., 1978) With a 6820 PIA, some 7404 buffers and a ZIF 16 pin socket together with a program KIM-1 can test IC's.

382. People's Computers 7 No 2 (Sept./Oct., 1978)
Zimmermann, Mark "Snooping With Your PET". A sophisticated Guide to PEEKing and POKEing around in PET.
Gaines, John "Apple Math". A math program for the Apple II.
Cole, Phyllis "SPOT" (Tips for PET trainers) - Discussion of the slow documentation for the PET. Program for lining up dollars/cents tabulations. PET listing conventions.

383. Conway, John "A Tape-to-Microcomputer-Hardware Interface Requires a Wealth of Micro-techniques". EDN 23 No 6 pg 101-110 (March 20, 1978). EDN project Indecomp tape interface hardware. Also the first hint of problems in interfacing Apple II with a PIA.

384. Hemenway, Jack E. "Add Floppies to Your Microcomputer to Form a Real Microcomputer System". EDN 23 No 12 pg 98-107 (June 20, 1978) re: May 20 EDN problems with tape hardware interfaces for Apple II. Also, EDN 23 discusses disk interface hardware. Final test to come.

385. Kilobaud Issue 23 (Oct., 1978)
Trageser, Jim "Budget System with KIM". How to expand your KIM system with ASCII keyboard, TVT-6 and associated software.
Ngai, Philip "Build a One-Chip Single Stepper". A debugging aid for home-brewed 6502 systems.
Kurtz, Robert L. "World of the Brass Pounders: Receive Morse Code the Easy Way". This Morse code reader is a good example of how the microcomputer can serve the radio amateur. Uses KIM-1.
Borland, D. "Financier/Mortgage with Prepayment". A pair of PET programs from Kilobaud's Instant Software line.
Beymer, Easton "Universal Number Converter". A program in PET BASIC for converting from one number base to another. Not only hex, decimal, octal and binary, but others.
Grossman,,Rick "Do It with a KIMSI". Use S-100 boards with the KIM. An evaluation.
Bishop, Robert J. "The Remarkable Apple II". A description and evaluation of the Apple II by a veteran 6502 programmer.

386. Palenik, Les "Formatting Dollars and Cents". Byte 3 No 10 pg 68 (Oct., 1978) This program for the PET rounds the monetary amount to the nearest cent and lines up the decimal points.

387. Bishop, Robert J. "Maze" Byte 3 No 10 pg 136-138 (Oct., 1978). This novelty program generates and displays a different maze about once a minute.

388. 6502 User Notes 12 (Nov., 1978)
Flacco, Roy "Scope Lunar Lander--final installment". The last part of a series incorporating graphics into the KIM-1 Lunar lander program.
Allen, Michael "TVT6 Etch-A-Sketch". A sketch routine for the KIM/TVT6 said to overcome the limitations imposed by a snowy screen during program execution.
Hooper, Philip K. "TVT6" - Some cheap, easy and helpful TVT6 hardware modifications.
Clem, D. "Expansion Decoding". One possible configuration of expansion decoding for KIM is said to be designed with TVT6 in mind.
Lewart, Cass and Lewart, Dan "TVT6 Remarks". Notes on improving the operation of the KIM/TVT6 system.

6502 User Notes 12 (Nov., 1978) con't.

Kushnier, Ronald "Notes on the TVT6". Information on memory expansion, the TVT6/KIM as a terminal, assembly of the TVT 6, the PVI-1K kit, KIM modification and use of the TVT6 with the Radio Shack keyboard.

Brachman, Michael "Suggestions for Running the TVT6". Includes a slight software mod to display pages 02,03,00 consecutively filling the whole screen (24 line x 64 character display.

Anon, "FOCAL" Staff evaluation of FOCAL for the 6502.

Latham, Don J. "Letter to the Editor". Comments on tape storage problems, KIMSI, XITEX, TVT6,Microsoft Basic, FCL-65, etc.

Rehnke, Eric "The Cheap Video Cookbook" - a book review. Very favorable review of Don Lancaster's latest book.

Martin, Timothy "KIM Interval Timers". Useful precautions in using the KIM interval timers are given.

389. MICRO, No 7 (Oct./Nov., 1978)

Auricchio, Rick "Breaker:An Apple II Debugging Aid". BREAKER is a software routine to manage Breakpoints, correctly resuming the user program after hitting a breakpoint

Watson, Allen III "MOS 16K RAM for the Apple II. Speed codes used by 16K Dynamic RAM manufacturers. The author advises against using 300 ns access time chips.

Creighton, Gary A. "PET Update". Discussion of the RND function, USR, Machine Language Storing in Basic, Save and Load, etc.

De Jong, Marvin L. "6502 Interfacing for Beginners; The Control Signals". The latest article in this series discusses the theororetical basis and progresses to hardware and a program for experimenting with control signals.

Shryock, William H., Jr. "Improved Star Battle Sound Effects". Further improvements based on the original article by Andrew H. Eliason in Issue no 6 of MICRO.

Green, J.S. "650X Opcode Sequence Matcher". A program that correlates and points to parallel sequences of opcodes, comparing the two sets and displaying the differences.

McCann, Michael J. "A Memory Test Program for the PET". Program is written in Commodore BASIC and occupies the lowest 4K of memory.

Rowe, Mike (Micro Staff) "The MICRO Software Catalog: IV" Ten more programs are reviewed in this continuing series.

Schwartz, Marc "Apple Calls and Hex-Decimal Conversion". How to access machine language routines by Calls and how to use Apple in helping make the hex-decimal conversion.

Dial, William R. "6502 Bibliography: Part VI". The 6502 literature continues to expand.

Dial, William R. "6502 Information Resources". A list of the magazines used in compiling the 6502 Bibliography and their subscription prices.

Powlette, Joseph L. and Wright, Charles T. "KIM-1 a Digital Voltmeter". Hardware and Software to convert the KIM-1 to use as a digital readout voltmeter.

Miller, Fred "Cassette Tape Controller". Control two tape units with your KIM.

Eliason, Andrew H. "Apple II High Resolution Graphics Memory Organization". A useful contribution toward better understanding of the Apple II HIRES Graphics.

DeJong, Marvin L., Riverside Electronics Design's KEM and MVM-1024: A User's Evaluation". A generally favorable evaluation of the KEM expansion board for KIM and the keyboard/video monitor board.

Sullivan, Chris "A Digital Clock Program for the SYM-1". This 24-hour clock program provides a good way for the new SYM owner to become familiar with the monitor subroutines.

Herman, Harvey B. "Peeking at PET's Basic". The PEEK function is used to look at the BASIC itself.

Tepperman, Barry "Kimbase". A program to convert from almost any number system to another. Contains many useful subroutines for multiplying, dividing and other uses.

390. Anon, "Tone Routine for Apple II". Southeastern Software Newsletter Issue No 3 pg 6 (Oct., 1978) The tone routine used with Apple Integer Basic can be used with Applesoft but must be relocated.

391. Haller, George "Storing and Recoving Data in Applesoft II". Southeastern Software Newsletter Issue No 2 pg 4 (Sept, 1978) Program for storing and recoving data in Applesoft II.

392. Call - APPLE 1 No 6
Williams, Don "Key Klicker Routine". A machine language program to provide a click each time your silent keyboard is punched
Anon, "Routine to Find Page Length". A routine to fill a page with repetitive material or to determine the length of a screen page of print statements.
Anon, "Printer Driver Fixes". A short program to prevent problems when using the Apple with a printer with more than 40 columns.
Anon, "Apple II Mini-Assembler". Discussion of the Apple II mini-assembler.
Aldrich, Darrell "Use of Color Mask Byte in HIRES". Brief description of this important aspect of Apple HIRES.
Anon, "Memory Map-Apple II with Applesoft Basic Loaded". Convenient tabulation of memory positions.
Anon, "List of Handy Calls". Sixteen calls are listed for the Apple II.
Apple Computer Staff "System Monitor". Discussion of how to get the most out of your Apple Monitor.
Huelsdonk, Bob "Memory Test". A test for Apple memory by loading each location with 55, testing, loading with AA and testing. Offending address will be shown.

393. Call - APPLE 1 No 7 (August, 1978)
Golding, Val J. "A Disk Utility Program". A program to record a group of programs on tape from a disk.
Backman, J. A. "Poor Man's HEX-DECIMAL-HEX Converter". With this table and a scratchpad, conversions are a snap.
Thyng, Mike "Basic File Handling". Discussion of the actual commands necessary to get data to and from diskettes using the Apple with a PERSCI disk drive.
Apple Computer Staff "System Monitor". Cassette I/O's, Memory Move and Verify, Debugging Aids, Single Stepping, Tracing, using the Apple II monitor.
Anon "Applesoft Zero Page Usage". Explanation of the functions residing in page zero of Apple II.
Huelsdonk, Bob "Routine to Print Free Bytes". Routine for Apple II with less than 32K memory.
Huelsdonk, Bob "A Patch for Double Loops". Discussion of precautions to use with with double loops in Applesoft on the APPLE II.
Apple Computer Staff "Loading Machine Language as Part of a Basic Program". Reprinted from Contact No 1, May 1978. Provides a way to include a machine language program within a Basic Program.

394. Call - APPLE 1 No 8 (Sept., 1978)
Aldrich, Ron "Convert". Program Loads Integer Basic Program from Disk, saves to a text file on disk, then executes that file in Applesoft II.
Thyng, Mike "Arrays". A description of the use of arrays with the Apple.
Chapman, Dan "Video Display Organization". A program to demonstrate the Video Display organization.
Anon "Routine to Save an Array". (reprinted from Apple Stems Vol 1 No 2 July, 1978) A routine to save both integer or floating point real numbers in an Applesoft II array.
Lam, S.H. "Monitor Commands from Basic." This routine allows execution of Apple Monitor commands from Basic with return to Basic.
Williams, Don "Linkage Routines for the Apple II Integer Basic Floating Point Package". A discussion of the Apple II ROM routines.
Hill, Alan G. "Return to TEXT from Graphics". A handy routine which permits the use of Control Y to return to Text from Apple Graphics.
Anon "Integral Data IP 125-225 Driver". A slight modification of the Apple Red Book teletype routine for use with the Integral Data printer.
Huelsdonk, Bob "Printer Driver Fixes". Protocol to use a printer with more than 40 columns with the Apple.

395. Call - APPLE 1 No 9 (Oct., 1978)
Cook, John B. "Applesoft Tone Routines". Relocation of the tone routines is necessary for use with Applesoft II on the Apple.
Scott, Michael M. "A Brief History of Apple". An interesting account by the President of Apple Computer Co.

Call - APPLE 1 No 9 (Oct., 1978) cont.

Anon, "Some Basic Entry Points". Various Call and JSR functions for Apple basic.
Huelsdonk, Bob "Sample File Handler". A program that demonstrates and will establish files for data handling, using the DOS on Apple.
Golding, Val and Williams, Don "Apple II Integer Basic: Interpretation of Memory". Tabular listing of pointers and tokens for Apple II Integer Basic.
Golding, Val "Applesoft II Tokens". Memory tabulation for tokens and pointers of Applesoft II Basic for the Apple.

396. PET Gazette 1 No 5 (Aug./Sept., 1978)
Anon, "PET Standards". Standards are suggested for writing PET programs, graphics, music listings, etc.
Louder, Mike "Dynamic Keyboard". Discussion of methods to change lines in Basic programs while the PET is running.
Richter, Mike "Data Files". Recommended procedures in preparing tapes for exchange.
Anon, "PET Memory Map". Map reprinted from the PET Paper.
Staebell, Jon "PET Hints". Miscellaneous hints for PET owners.
Modeen, Roger L. "Easy Auto Answer/Originate Modem". Modem for the PET.
Clark, Ken "Proposed Temporary Local Standard for Low Speed Data Exchange by Modem". Protocals for data exchange.
Cumberton, Dennis "File Manager". A program for reliable data reading and writing with the PET.
Bendoritis, Bill "Renumber". A renumbering program for the PET.

SYM-PLICITY

ENGINEERED SPECIFICALLY FOR THE SYM-1 MICRO COMPUTER
- Easy Access for Adjustments
- Room for Expansion
- Protects Vital Components

EASILY ASSEMBLED
- Absolutely No Alteration of SYM-1 Required
- All Fasteners Provided
- Goes Together in Minutes

MADE OF HIGH IMPACT STRENGTH THERMOFORMED PLASTIC
- Kydex 100*
- Durable
- Molded-In Color
- Non-Conductive

AVAILABLE FROM STOCK
- Allow Two to Three Weeks for Processing and Delivery
- No COD's Please
- Dealer Inquiries Invited

ATTRACTIVE FUNCTIONAL PACKAGE
- Professional Appearance
- Popular "Data Blue" Color
- Improves Man/Machine Interface

TO ORDER: 1. Fill in this Coupon (Print or Type Please)
2. Attach Check or Money Order and Mail to:

NAME ______________________________

STREET ______________________________

CITY ______________________________

STATE ______________ ZIP ______________

Please Ship Prepaid _____ SSE 1-1(s)
@ $36.75 each
California Residents Please Pay
$39.14 (Includes Sales Tax)

enclosures group

753 bush street
san francisco, california 94108

* TM Rohm & Haas

Patent Applied For